

AD-A087 562

SIMULATION TECHNOLOGY INC NASHVILLE TN
INCREASED OPERATIONAL FACILITY OF THE IRSS.(U)
MAY 80 R J HANCOCK

F/G 17/9

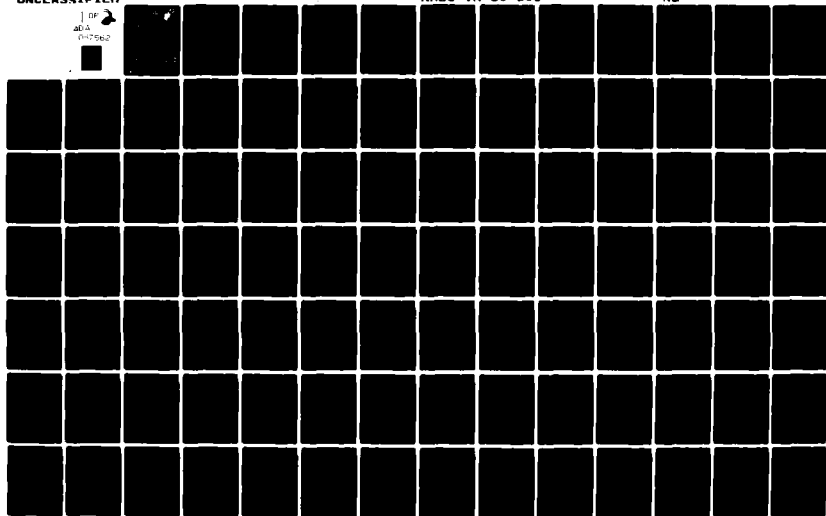
UNCLASSIFIED

RADC-TR-80-150

F30602-79-C-0043

NL

1 OF
404
017562



LEVEL II

RADC-TR-80-150
Final Technical Report
May 1980



ADA 087562

INCREASED OPERATIONAL FACILITY OF THE IRSS

Simulation Technology Incorporated

Robert J. Hancock

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

DTIC
ELECTE
AUG 6 1980
S D

DDC FILE COPY

80 8 4 043

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-80-150 has been reviewed and is approved for publication.

APPROVED:



STANLEY D. DAVIS
Project Engineer

APPROVED:



FRANK J. REHM
Technical Director
Surveillance Division

FOR THE COMMANDER:



JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (OCDR) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER RADC-TR-80-150	2. GOVT ACCESSION NO. AD-A087	3. RECIPIENT'S CATALOG NUMBER 562	
4. TITLE (and Subtitle) INCREASED OPERATIONAL FACILITY OF THE IRSS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report	
7. AUTHOR(s) Robert J. Hancock		6. PERFORMING ORG. REPORT NUMBER N/A	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Simulation Technology, Inc. 390A Haywood Lane Nashville TN 37013		8. CONTRACT OR GRANT NUMBER(s) F30602-79-C-0043	
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (OCDR) Griffiss AFB NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45061640	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		12. REPORT DATE May 1980	
		13. NUMBER OF PAGES 100	
		15. SECURITY CLASS. (of this report) UNCLASSIFIED	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same			
18. SUPPLEMENTARY NOTES RADC Project Engineer: Stanley D. Davis (OCDR)			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Radar Modelling Simulation			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This effort is concerned with development and implementation of a revised software architecture that will permit the previous Interactive Radar Simulation System to be utilized in an easier and more productive manner. All of the previous capabilities that reside in the actual models remain intact, however the method of using them has been significantly improved.			

DD FORM 1473
1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

-78 394-535-11

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

<u>SECTION</u>	<u>TITLE</u>	<u>PAGE</u>
1	INTRODUCTION	1
2	SUMMARY	3
3	BACKGROUND	5
3.1	General	5
3.2	The Role of Models in System Design	5
3.2.1	Mathematical Analysis	6
3.2.2	Simulation Procedure	7
3.3	Radsim Program Structure Evolution	7
3.4	Initial IRSS Configuration	14
3.4.1	RADSIM Computer Program	14
3.4.2	IRSS Weaknesses	18
3.4.3	Potential Improvements	20
4	IRSS DESCRIPTION	23
4.1	General	23
4.2	Structural Description	23
4.3	Operational Description	25
4.3.1	Job Setup Procedures	25
4.3.2	Error Checking and Recovery	27
4.3.3	General Information Retrieval	28
4.4	Control Flow	28
4.5	Subsystem Descriptions	30
4.5.1	Interactive Translator Subsystem	30
4.5.2	HELP Processor	35

<u>SECTION</u>	<u>TITLE</u>	<u>PAGE</u>
	4.5.3 Waveform Processing Subsystem	35
	4.5.4 User Aid Processor	37
5	CONCLUSIONS AND RECOMMENDATIONS	39
	5.1 Conclusions	39
	5.2 Recommendations	40
6	REFERENCES	43

APPENDICES

A	AN EXAMPLE OF INTERACTIVE EXECUTION	A-1
B	AN EXAMPLE OF USER AID PROCESSOR EXECUTION	B-1
C	AN EXAMPLE OF SIMULATION RUN FILE EXECUTION	C-1
D	LIST OF ACRONYMS	D-1

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist.	Avail and/or special
A	

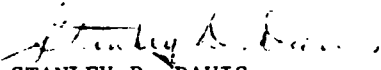
DTIC
ELECTE
S AUG 6 1980 D
D

LIST OF FIGURES

<u>NUMBER</u>	<u>TITLE</u>	<u>PAGE</u>
3-1	RADSIM-B Structure and Operational Procedure	9
3-2	RADSIM-C Structure and Operational Procedure	12
3-3	RADSIM-C/DUIS Structure and Operational Procedure	15
3-4	Example of GCDS Job Definition File	17
4-1	IRSS Structural Block Diagram	24
4-2	Block Diagram of Program Control Flow	29
4-3	IFE Functional Block Diagram	32
4-4	Example of IFE Information Retrieval from RADSIM Data Base	33
4-5	WPS Functional Block Diagram	36

EVALUATION

1. The objective of this effort was to modify the software structure of RADC's existing Interactive Radar Simulation System (IRSS) to make a major improvement in the ease of using the system and to shorten the turn-around time in getting plotted output results. These marked improvements will be reflected in greater use by RADC engineers in areas of ECCM, new radar parametric design studies and determination of problem areas or causes in existing radar systems.
2. This effort is part of RADC TPO-4B.


STANLEY D. DAVIS
Project Engineer

SECTION 1

INTRODUCTION

In designing a radar system the goal is to combine various components in such a way that a certain capability is created. In a sense the problem of building a system to satisfy a requirement is a problem of implementing a model. That is to say, for any object constructed by man the idea came first, then the object appeared after a way was found to implement the idea. In the more formal language of systems engineering, ideas are mental models. Initially in the design process we envision a perfect system composed of flawless components (a Perfect Model). Then, faced with reality, we back away and accept compromises in performance. Some are easy to accept; others are more difficult. All the while we must ensure that the cumulative effect of these compromises will not render the total system useless.

In developing a complex system the design and development process involves a progressive refinement of models, especially with regard to subsystem specification. When large complex systems are involved it is desirable to implement a simulation model of the entire system based on the design of each individual subsystem. In this way the system designer can determine marginal areas in his design and correct them. Simulation models can be used to optimize designs and reduce some of the risks involved in system development.

Over the past several years a comprehensive Radar System Simulation Model (RADSIM) computer program has been available for use in radar system analysis and performance prediction. The RADSIM program began as a batch job executed on an IBM 360 computer. This original program was later modified and installed in the RADC GE 635 computer. As a result of subsequent contractual efforts the capabilities of the program were expanded to include an exoatmospheric chaff model, target model, propagation effects, ECM, and more advanced waveform phase coding techniques. In addition, the operational procedure was changed so that the program could be executed as a remote batch job using the CARDIN facility available under GCOS Time Sharing System (TSS). The internal architecture of RADSIM remained essentially unchanged.

In using RADSIM on the RADC computer it was noted that the most time consuming aspect of a simulation was the evaluation of the results. The second most time consuming aspect was the preparation of RADSIM Simulation Input Files (SIF's). To help alleviate these shortcomings the Dedicated User Interface Subsystem (DUI) was designed and implemented. The DUI provides a good on-line plotting capability for supporting RADSIM and other GCOS computer programs. It does not, however, provide sufficient aid to the user in the setup of simulation jobs. The effort described herein was directed toward the implementation of the necessary support software to simplify the user's task in the setup of radar simulation jobs. The effort documented herein is the final stage in the evolution of the radar simulation from a pure batch job (RADSIM) to a highly interactive design tool (IRSS).

SECTION 2

SUMMARY

This report documents a study and investigation directed toward modifying the Interactive Radar Simulation System (IRSS) controlling software structures to simplify the user-IRSS interface functions. In performing this effort the RADSIM computer program, formerly run under remote batch, was converted for execution under time-sharing in a truly interactive manner.

In performing this effort only the controlling software structure was modified. The characteristics of the simulation modules were not changed. Therefore, the module descriptions which have been documented in an earlier report (Ref. 1) are still valid. In addition, the Graphics Control Subsystem (GCS) and the DUIS were unchanged and also have been documented in earlier reports (Ref. 2 & 3). In this report the information considered in arriving at a modified IRSS design is discussed in Section 3. In Section 4 the current configuration of the IRSS is described.

The operational procedure for the IRSS is documented in the User's Manual which is contained in a separate volume. In Appendices A, B, and C of this report examples are provided of simulations performed using the IRSS. These examples were chosen to illustrate: interactive operation, computer directed simulation setup, and operation from a simulation RUN file. Appendix D contains a list of acronyms.

SECTION 3

BACKGROUND

3.1 GENERAL

In this section the various elements of background information which were considered in arriving at the implemented design are discussed. First, the role of models in system design is discussed and the stages in the design process where simulation is cost effective are identified. Second, the evolution of the RADSIM computer program is reviewed. Third, the configuration of the IRSS at the beginning of this effort is described and its capabilities and weaknesses are evaluated. Fourth, potential improvements are described which would have increased the utility of RADSIM.

3.2 THE ROLE OF MODELS IN SYSTEM DESIGN

The process of designing a system is strongly dependent on the use of models. The definition of model used here is: "an abstract object which describes or represents a physical object". The design of a complex system can be thought of as the successive refinement of a model. This process will be briefly outlined as follows:

1. A desired operational capability is defined. This normally consists of an objective and a set of groundrules.
2. In order to satisfy this capability a number of possible solutions is explored.
3. The solutions having the most promise are defined in more detail in order to form a basis for comparison.
4. By comparing the models of competing solutions a conclusion is reached as to which is the preferred approach to satisfying the objective.
5. In order to build the selected system the model must be further refined to break it down into its component parts. Ultimately, a set of detailed specifications for the component parts, usually

subsystems, must be generated so that they can be built or purchased.

6. Once the characteristics of the component parts of the system have been defined then a system model is built from the bottom up and system performance estimated.
7. The specifications of the subsystems are refined as necessary to optimize the ratio of cost + risk to performance.
8. Specifications are frozen and the component parts of the system are purchased or built.
9. An experimental model of the system is configured, tests are performed, and the subsystem specifications are refined as necessary.

Depending on the complexity of the system involved, various steps in the process described above may be omitted. From the standpoint of minimizing design risk the procedures described in Steps 6 and 7 are crucial. In the past Step 6 often has been nothing more than the formalization of the results from Step 5 in the form of a system Accuracy Control Document (or performance analysis). In order to truly minimize risk, the evaluation performed in Step 6 should be an independent evaluation. Specifically, it should be a detailed model which accurately represents the internal behavior of each subsystem. These refined models typically have involved either mathematical analysis or simulation.

3.2.1 Mathematical Analysis

In mathematical analysis the goal is to combine the various models using algebraic techniques and arrive at a single equation or set of equations which will give system performance as a function of various system parameters. Providing that no simplifying assumptions, approximations, etc. are made, this approach will lead to an accurate estimate of system performance. Unfortunately, radar systems are loaded with subsystems which make life difficult for the mathematical analyst, e.g., nonlinear effects such as receiver compression and saturation, analog to digital converters, and digital processor arithmetic truncation. In addition, the usual mathematical analysis techniques are heavily biased toward using Gaussian distributions for everything. As long as the real world

can be characterized by Gaussian distributions, mathematical analysis solutions are obtainable and sometimes are accurate representations of the system analyzed. An advantage of analysis solutions is that the resulting equation or equations can be solved in a straightforward manner to produce radar system performance curves.

3.2.2 Simulation Procedure

In obtaining solutions via simulation the goal is to cause a digital computer or some special purpose simulation hardware to behave in the manner prescribed by the various subsystem models. In this way the response of the radar system to a particular environment can be accurately predicted. The advantage of simulation is that radar subsystems can be accurately modeled, even down to the bit level for digital signal processors. Empirical data can easily be used, e.g., antenna patterns, receiver transfer functions, etc. Simulation procedures are not biased toward any particular probability distribution. Therefore, assumptions about Gaussian statistics are unnecessary. The use of simulation and the interpretation of results does not require the mathematical sophistication which is needed to successfully produce closed form mathematical solutions. Simulation does have the disadvantage that, in general, it does require more computer time than the evaluation of closed form solutions obtained via mathematical analysis. This is especially true when dealing with stochastic processes for which one desires to generate statistics that are representative of an entire population. It should be noted that typically mathematical analysis requires more manhours than does the setting up of a simulation once the simulation software has been developed.

3.3 RADSIM PROGRAM STRUCTURE EVOLUTION

In this subsection the development of RADSIM is discussed. It should be noted that this description of the RADSIM program evolution is concerned with structural changes that effect user efficiency and perception of RADSIM. The description contained herein does not address itself to the development of simulation and analysis modules which make up the core of RADSIM. These modules are virtually independent of the user interface structure.

RADSIM began as a batch job that was run using punch card input media only. In setting up the program Input/Output (I/O) structure an attempt was made to minimize the amount of user input data that was required. This was done for two reasons. First, the probability of making a typographical error is proportional to the number of characters keypunched. Second, the keypunching of cards is a tedious, time consuming process that should be minimized if possible. Therefore, a rather cryptic simulation command structure was developed using a combination of simulation control words (SCW) and module reference numbers (MRN). In this way only nine characters (SCW = 6 CHAR and MRN = 3 CHAR) on one punch card were required to cause execution of a RADSIM module. Minimization of the number of characters per punch card as well as the minimization of the total number of punch cards was the primary goal. For specifying module parameter data NAMELIST I/O was used because it is essentially self-documenting since parameter names are paired with their values, e.g. entries are of the form: PARAM=10.0,IFLG=1,etc. The program was run for a few years in this form on both GE 635 and IBM 360 computer systems.

A block diagram of the RADSIM-B structure and operating procedure is shown in Figure 3-1. In this figure the user is shown in two ways. The user appears at the top as an operator in the block entitled, "User Specification of Problem." The user also appears at the bottom as an analyst in the block entitled, "User Evaluation of Results." The procedure utilized in running this version of RADSIM is as follows: First, the user determines the necessary input data that is required to run the simulation model. This input data is given the name Simulation Input File (SIF). Second, the SIF is keypunched. Third, this card deck is combined with the necessary Job Control Language (JCL) to form the Job Definition File (JDF) and this is then given to the computer operator for execution as a batch job.

Once program execution begins, the SIF is processed by the input processor portion of the program which takes the input data and converts it into a compact form for subsequent use by the simulation and analysis modules. The reason for doing this is that I/O format statements for reading in the user's data and the necessary pre-processing of that data require approximately 20K of core. In order to minimize the amount of core required to run a simulation job it was decided to separate the job into two load modules. In this procedure the total JDF is processed by

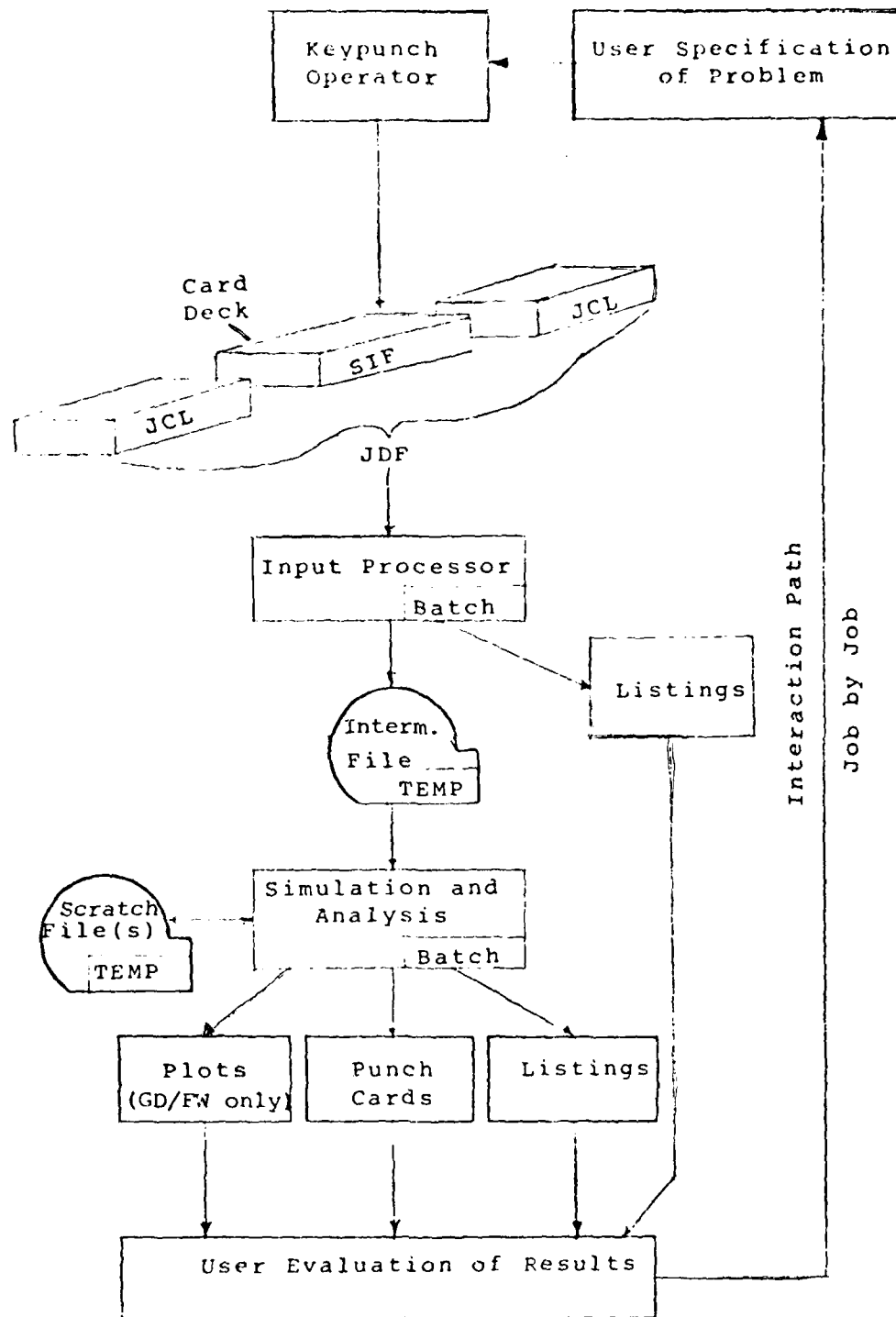


Figure 3-1 RADSIM-B Structure and Operational Procedure

the input processor and the output is placed on an intermediate file. A listing is produced of all the user's input data plus any error diagnostics.

Next the simulation and analysis load module is executed. Its input data is from the intermediate file loaded by the input processor. It also can make use of scratch files. All of the output data from the simulation and analysis load module is placed on an output file. This output file is subsequently printed on a line printer or used in an off-line plotter. The user then evaluates the output produced by the simulation and determines if the desired results were achieved or if modifications need to be made and another job run. If so, this path is shown in the figure going from the user evaluation back up to the user as an operator. This interactive path is rather long and is basically done on a job-by-job basis. Typically, the length of time required between the submission of a job and the evaluation of the results was on the order of half a day. It should be noted in this figure that both the input processor and the simulation and analysis load modules were executed as batch jobs. Also all files generated were temporary files, i.e., they were purged from the system as soon as the batch job was completed. Therefore, all input data and output data existed only as punch cards, paper listings, or plots.

After the time sharing capability of the GE 635 had sufficiently matured, it was decided to try converting RADSIM from a pure batch processing mode to remote batch execution. The initial goals were to avoid having to handle boxes of punch cards and to allow program check out and debugging to be done from remote terminals. Preliminary results were good and therefore the whole RADSIM program was put on permanent disc storage (PRMFL) in the GE 635 at RADC. Since then all access to, modification of, and use of RADSIM was done via time share (TSS). Initially, the program structure remained the same as it was in batch processing. Then some modifications were made to minimize the size of printouts (reduce job output to an absolute minimum). This was desirable since most remote terminals at that time operated at 30 characters per second or less. Therefore, printing long listings was time consuming. In batch jobs (the old way) the output was produced on a line printer and therefore one worried little about compactness. The initial approach to input was still good since it minimized connect time to the host computer and the amount of typing required of the user. Therefore there was no attempt to modify it. The biggest problem was

to efficiently prepare simulation output data for evaluation by the user. The procedure for obtaining plots of the output data was time consuming since it involved punching plot data on cards and subsequently plotting the data on an HP9820 desk calculator.

A block diagram of RADSIM-C is shown in Figure 3-2. In this figure the GE 635 text editor replaces the keypunch operator. The user now creates his JDF by using the text editor. When he is ready to run a job, the JDF is submitted to the batch processor via the remote batch system (CARDIN).

Functionally, the input processor operates in the same manner as before, that is, the SIF is read in, processed, and an intermediate file generated. It differs in that instead of directly producing a listing its output goes into a JOUT file so that the user can subsequently look at it via a remote terminal, if desired. After the input processor load module has completed its processing the simulation and analysis load module is loaded and executed. The input data for this load module is the intermediate file generated by the previous execution of the input processor. The simulation output data is written to a PRMFL which was allocated by the user. The system output is again written to a JOUT file for subsequent access by the user from a remote terminal. The output data files can, at the option of the user, either be punched on card decks or listed. The data punched on card decks normally was used to generate plots on an HP desk calculator. The user had the option of determining the plot axis scales that were used in making the plots, and therefore, had some control over the data that was plotted independent of the execution of RADSIM. It should be noted that the output data file and the JDF were both stored in PRMFL's, and therefore, remain in the system permanently until purged by the user. The intermediate file and the scratch files were still temporary and existed only as long as the batch job was resident in the machine. The user is still shown at the bottom as an analyst evaluating the output data. Based on his decisions, he could make changes to his specification of the problem and rerun the job.

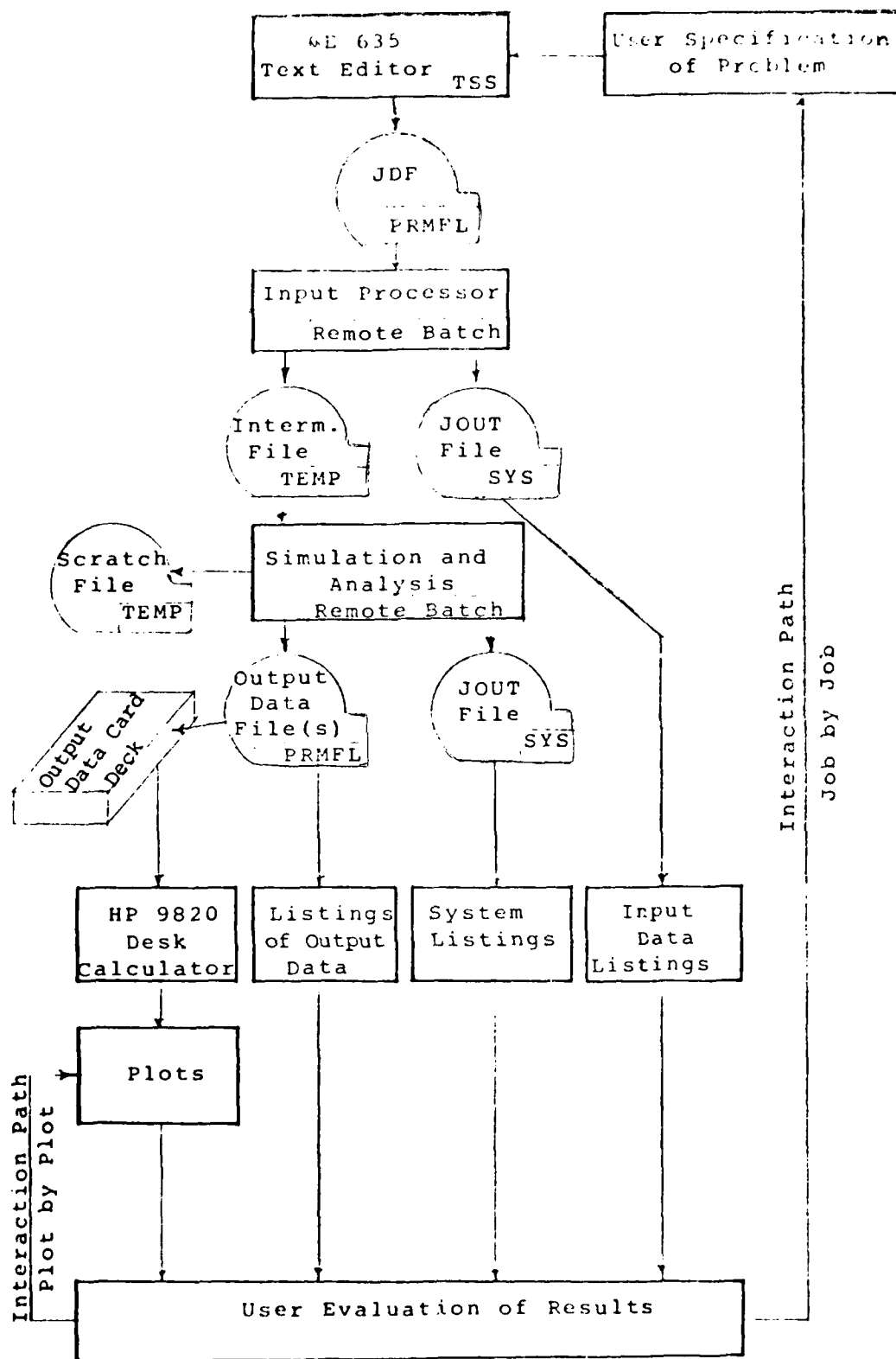


Figure 3-2 RADSIM-C Structure and Operational Procedure

In this case, the turnaround time from submission of a job to its output was on the order of 30 minutes or less. The big problem with this technique was the time delay involved in getting cards punched and transporting them to the desk calculator for plotting. Even though the execution of the RADSIM program had been significantly improved the overall system performance was still rather poor requiring roughly two hours per job if plots were desired.

This problem of efficiently presenting simulation output data to the user was studied and the conclusion reached was that a remote job entry processor was needed which had a plotting and hardcopy generation capability. Subsequently, the DUIS was designed, developed, and integrated with RADSIM. The combination of the DUIS and RADSIM was referred to as the Interactive Radar System Simulation System (IRSS) and is discussed further in the next subsection.

3.4 INITIAL IRSS CONFIGURATION

In this subsection a description is provided of the IRSS as it existed when the effort documented herein was initiated. During development of this system emphasis was placed on the efficient production of plots from simulation results. The secondary goal was to improve the input side of RADSIM by providing DUIS resident look-up tables for the user. This, however, was found not to be feasible for a system which did not have a disc drive for storing the large amounts of reference data needed. Therefore, this effort was successful mainly in improving the output side of RADSIM.

In Figure 3-3 the block diagram is shown of the IRSS which resulted from adding the DUIS to RADSIM. The program is still executed as a batch job. The software which makes up the RADSIM program itself is the same in this figure as in the previous figure. In this diagram the primary user data entry device and also data retrieval device, is a Tektronix 4014 graphics display. This display is used to input the data to the DUIS input processor which in turn prepares the job input files for the RADSIM program. This input process can be executed as a stand alone program in the DUIS, or can be used in conjunction with the editing capabilities of the H6180. On the output side of the simulation, the output file is accessed by the DUIS through an output processor which is resident in the H6180.

The output processor and the DUIS graphics control processor work together in order to transfer data from an output file, loaded by a simulation, to the DUIS for subsequent plotting. The program can transfer data from the H6180, immediately plot it, and produce hard copies if desired by the user. One of the requirements in developing the DUIS was that the RADSIM I/O structure could not be changed. This was done to ensure that the program could be run from any remote terminal.

3.4.1 RADSIM Computer Program

In this paragraph the RADSIM computer program is discussed with emphasis on the program architecture and user input data requirements.

3.4.1.1 RADSIM Architecture

The RADSIM computer program was divided into two segments which were executed as separate load modules in

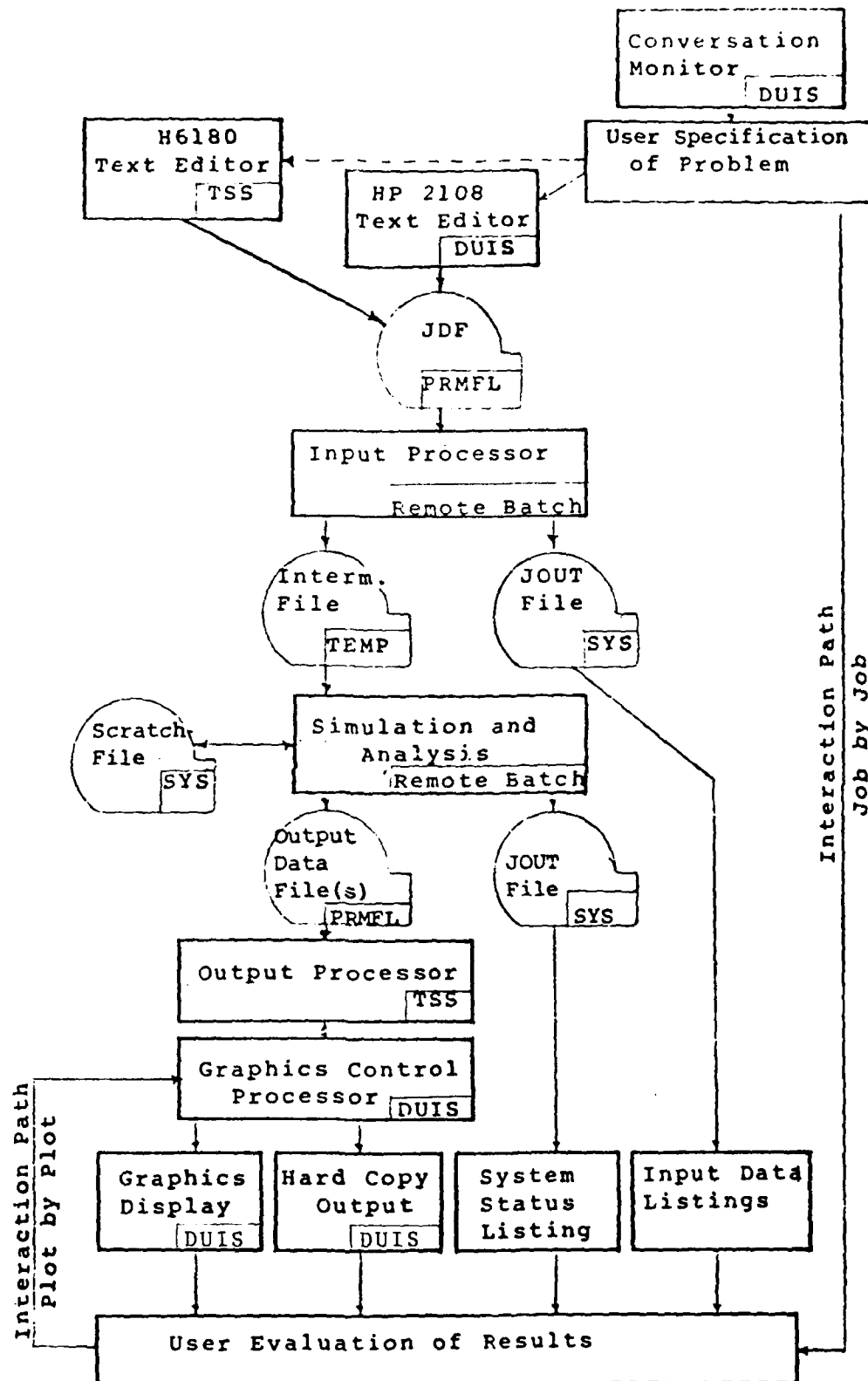


Figure 3-3 RADSIM-C/DUIS Structure and Operational Procedure

the H6180. The first segment processed the Simulation Input File (SIF) that defined the simulation to be performed. This preprocessing consisted of generating a Simulation Module Execution Sequence Table (SMEST) and placing data in the Module Parameter Table (MPT) and in the Module Array Table (MAT). The contents of the SMEST, MPT, and MAT were stored on a temporary file for subsequent use by the simulation segment. In addition, the clutter model initialization was performed if the clutter model was to be used in the simulation.

The second segment performed the actual simulation. The selection and order of execution of the simulation modules was determined by the SMEST. Each combination of simulation module and waveform storage assignment was specified by a Module Reference Number (MRN). The available waveform storage areas were designated by the symbolic names XT, YT, XA, and XB. These waveform storage arrays were core resident which severely limited the expansion potential of the program. For any module if more than one combination of waveform storage assignment was desired then additional MRN'S were assigned.

Both activities of RADSIM were core resident. The load module for the second segment required approximately 70K of memory. This large load module size resulted from the fact that all simulation modules to be used in performing a job were loaded into core and remained there throughout the simulation even though only one at a time was actually being used.

3.4.1.2 GCOS Job Definition File

In order to define a GCOS batch job one must communicate with GCOS via control cards. The set of GCOS control cards and simulation input data are collectively referred to as the Job Definition File (JDF). An example of a JDF is shown in Figure 3-4. The user provided SIF processed by the first segment is shown following the DATA card. This input data will be discussed further in paragraph 3.4.1.3. It should be noted that most of the cards which make up the JDF are required to define the load module. The only cards which were truly dependent on the user are the IDENT card which identifies the user to the operating system and the PRMFL card(s) to be used in storing plot data. Through the use of SELECT cards the user could, to some extent, tailor the simulation so that only those modules he required were included in the load module.


```

20$:IDENT:BECAVU01,HANCOCK ,651211040063,JOBNAM
30$:OPTION:FORTRAN
40$:SELECT:BECAVU01/MAINSJR/OLODRX
50$:SELECT:BECAVU01/SUPORTSJR/OSUPRT1
60$:SELECT:BECAVU01/SUBSYSSJR/OPHENC
70$:SELECT:BECAVU01/ENVR$JR/OCLINT
80$:SELECT:BECAVU01/SUPORTSJR/OARSIN
90$:EXECUTE
100$:LIMITS:10,30K,0,5K
110$:FILE:02,X1S,10R
120$:FILE:11,X2S,10R
130$:DATA:05

```

Simulation Input File

```

470$:OPTION:FORTRAN
480$:SELECT:BECAVU01/MAINSJR/OSIMEX
490$:SELECT:BECAVU01/SUPORTSJR/OPLOTS
500$:SELECT:BECAVU01/SUPORTSJR/OSUPRT1
510$:SELECT:BECAVU01/SUPORTSJR/OMISC1
520$:SELECT:BECAVU01/SUPORTSJR/OMISC2
530$:SELECT:BECAVU01/SUPORTSJR/OSUPRT2
540$:SELECT:BECAVU01/SUPORTSJR/OZFFT
550$:SELECT:BECAVU01/SUPORTSJR/OARSIN
560$:SELECT:BECAVU01/SUBSYSSJR/0XMTR
570$:SELECT:BECAVU01/SUBSYSSJR/ODFIN
580$:SELECT:BECAVU01/ENVR$JR/OTGCL
590$:EXECUTE
600$:LIMITS:10,65K,0,20K
610$:PRMFL:04,R/W,L,BECAVU01/DSTOR$JR/FILE1
620$:FILE:02,X1R,10R
630$:FILE:11,X2R,10R
640$:ENDJOB

```

Figure 3- 4 Example of a GCOS Job Definition File

3.4.1.3 Simulation Input File

The RADSIM SIF contains the simulation control and module parameter data. The simulation control cards determine which simulation modules are used and their order of execution. Each module has its own unique MRN or set of MRN's. Multiple MRN's are assigned to a module if more than one assignment of input and output waveform storage is required. For example, the module XYTODB has two MRN's: 104 and 108. If MRN = 104 is selected then the arrays XT and YT are processed and the result placed in XA. If MRN = 108 is selected then the same input arrays are processed but the result is placed in the array XT.

The specification of the input data for a module is done via namelist data card(s) which follow the simulation control card that selected the module.

3.4.2 IRSS WEAKNESSES

In this subsection the various weaknesses in the IRSS are discussed. These are grouped into three categories: Operational or general, DUIS, and RADSIM problems.

3.4.2.1 Operational Problems

In order to run a RADSIM job, a JDF was set up which contained a large number of GCOS control cards. For small simulation jobs the control cards actually outnumbered the SIF cards. This, therefore, represented an overhead task that the user must perform which was not directly related to radar system design. Because the RADSIM computer program was executed under remote batch there was no mechanism whereby an error made by the user could be corrected in a timely manner. For a batch job the only recourse when an error was detected was to print some diagnostic for the user and abort the job. Through CARDIN each job normally required up to 30 minutes or so to run. This means the turnaround time on each user error was quite long. In setting up a simulation job the user must specify the data parameters required by each module to be executed. These parameters were identified via FORTRAN symbolic names in the namelist data cards. The problem was that users often had difficulty relating these symbolic names to corresponding radar subsystem parameters. This parameter identification problem was to some extent alleviated through the use of module cross reference tables and parameter tables. These tables, however, were rather large and time consuming to use.

3.4.2.2 DUIS Problems

The DUIS as currently configured uses paper tape for peripheral storage. This has a number of drawbacks. One, paper tape is not a reusable media form and therefore is quite expensive. Two, I/O transfers with paper tape are relatively slow, i.e. 75 characters per second for output and 300 characters per second for input. Three, paper tape is not very reliable.

The HP 2108A minicomputer used in the DUIS is configured with 32K words of memory which is the maximum amount of memory that can be addressed without the use of mapping. Fortunately, all application programs developed to date, except the block diagram generator (BLDIAG), easily fit into the available memory.

The DUIS cannot support software development using FORTRAN IV since the FORTRAN IV compiler requires the use of a disc drive. This restriction to FORTRAN II places an additional burden on the programmer. Also, the FORTRAN II compiler does not make the most efficient use of available memory.

The goal of the BLDIAG program was to provide the user with an aid in the setup of the RADSIM jobs, specifically the automatic generation of the SIF. The BLDIAG program fell short of this goal for two reasons: One, the software required to properly do the job could not be "crammed" into the HP2108A. Two, an associated problem is that no disc storage was available and therefore the RADSIM cross reference tables and module parameter tables had to be restricted in size so that they could be stored in the computer main memory.

3.4.2.3 RADSIM Problems

Probably the biggest problem with RADSIM was that there was no mechanism for the user to correct his mistakes in a timely manner. Any error, no matter how minor, could ruin a simulation job.

The RADSIM program used an architecture which allowed the user to select only certain prestructured argument list combinations for each module. The user communicated his selection of module/storage list through the use of MRN's. This made the job setup difficult for the user because he

must reference a table to find out what the various numbers mean.

The RADSIM program was structured such that the whole load module was resident in core during the simulation activity. This resulted in a rather large load module. Because the waveform storage arrays were core resident the user was severely restricted on the length of the waveforms to be processed and the number of waveforms which could be used in a simulation.

3.4.3 POTENTIAL IMPROVEMENTS

In this subsection various ideas are described that were considered in order to improve the usefulness of the IRSS.

3.4.3.1 Operational Changes

From an operational standpoint the biggest problem faced by the user is remembering the meaning of various symbolic names, the I/O characteristics of the modules, input data requirements, etc. To alleviate this problem all of the RADSIM reference material should be incorporated into a data base which is disc resident. This data base should be easily accessible by the user from a terminal. Another potential operational change is the elimination of all of the GCOS control cards which must be prepared by the user.

3.4.3.2 DUIS Changes

The major problems with the DUIS are related to hardware, mainly the dependency on paper tape. The addition of flexible disc drives to the DUIS would provide an efficient, reliable means of storing and retrieving data. A secondary problem is the limited memory of the DUIS computer. To alleviate this at least 32K more memory and a dynamic mapping capability could be added. In addition, the DUIS operating system could be upgraded to make maximum benefit of the disc drives and memory mapping. Unfortunately, none of the recommended changes for the DUIS were funded under the effort documented in this report.

3.4.3.3 RADSIM Changes

Various aspects of the RADSIM computer program are "carry-overs" from its origin as a batch job. The following changes could be made to improve the usefulness

of the RADSIM program:

1. Convert the simulation activity to a link overlay to reduce the load module size
2. Store the RADSIM load module on a disc file and make it accessible to any user via TSS
3. Modify the technique for specifying modules so that module names rather than numbers (MRN's) are used. Also, the module I/O linkage should be directly specified in terms of array symbolic names.
4. Replace the data preprocessing activity with an interactive procedure.
5. Modify RADSIM so that waveforms are disc resident rather than core resident. This will provide the user with more flexibility and reduce the load module size.

SECTION 4

IRSS DESCRIPTION

4.1 GENERAL

In designing any tool it is difficult to compromise between the desires and capabilities of the novice, occasional, and experienced users. The novice must be guided through the procedure in order to familiarize him with it. The occasional user has a good idea of what he wants to do but can't remember exactly how to go about it. He therefore may commit some procedural error. The experienced user desires complete flexibility.

In modifying the IRSS two operating modes were implemented: user directed and computer directed. In the user directed mode the user has total flexibility and he can combine modules in any way he sees fit to form a simulation. This mode was intended for use by the experienced and occasional user only. In the computer directed mode the user is required to answer a sequence of questions. Based on the user's answers the IRSS sets up a Simulation RUN File (SRF) for the user. This mode is intended for use by the novice primarily. Its purpose is to help familiarize potential users with the capability of the simulation.

In this section the structure of the IRSS, the control flow, and the subsystems which form the IRSS are discussed.

4.2 STRUCTURAL DESCRIPTION

The structural block diagram of the IRSS is shown in Figure 4-1. In this figure PRMFL is used to indicate permanent storage of data on disc. SYS indicates that temporary storage is obtained from the operating system. TSS indicates that a subsystem is executed under H6180 TSS YFORT. The structure of the IRSS has been divided into three processes. The input process consists of selecting a module for execution and providing the necessary input data. The simulation process consists of executing the appropriate simulation module. The output process involves generating plots of simulated waveforms.

In the input process the Interactive Translator Subsystem (ITS) is used to process user commands and to

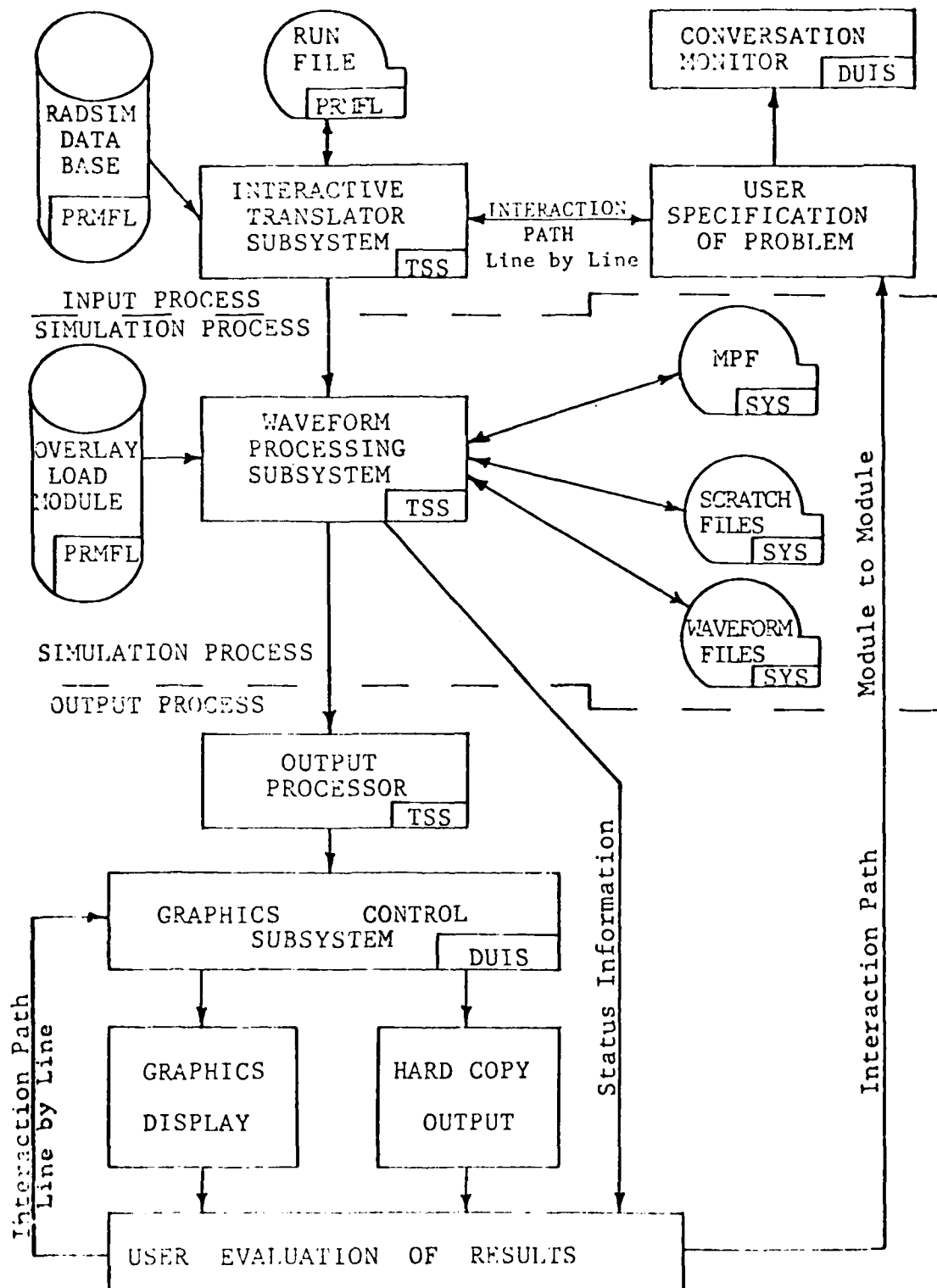


Figure 4-1 IRSS Structural Block Diagram

prepare all information required to execute a simulation module. The ITS can access the RADSIM data base in order to provide information to the user. The user communicates with the ITS on a line by line basis. If the user makes an error it is immediately brought to his attention. A conversation monitor feature is incorporated so that a permanent copy of the user's transactions with the computer can be generated. The RADSIM data base and the ITS are resident in the H6180. A facility has been provided such that the user commands and parameter specifications can be stored in a SRF. In that way a permanent copy of these inputs can be created for use at a later time.

In the simulation process the Waveform Processing Section (WPS) is structured as a link overlay and is resident in a PRMFL. The waveforms are resident in disc files also. The required memory for the load module is 34K. This small load module size has significantly improved job turnaround time and has reduced job costs. The Multiple Pass File (MPF) capability allows a simulation procedure to be rerun with simple parameter changes in order that parameter sensitivity analyses can easily be performed.

In the output process plot data are passed to the Graphic Control Subsystem (GCS) for plotting. The user can replot data on line and make hard copies as required. This portion of the IRSS is essentially unchanged from the previous version.

4.3 OPERATIONAL DESCRIPTION

The main consideration in modifying the IRSS was to minimize or otherwise simplify the inputs required from the user. To this end no GCOS control cards must be prepared since batch processing has been eliminated. The input requirements were simplified by implementing computer aids in job setup, a computer resident data base, and several levels of error checking.

4.3.1 Job Setup Procedures

In this paragraph the two procedures for setting up simulation jobs are described. All of the procedures are designed to rely heavily on information contained in the RADSIM data base.

4.3.1.1 User Directed Job Setup

Under this procedure the user is in control. He tells the computer which modules are to be executed, their parameters, the order of execution, and their I/O lists. When operating under this procedure the simulation is performed in a totally interactive manner. As the user specifies each module it is executed and the resulting output can be immediately plotted at the user's option. All user I/O is handled through the ITS which performs error checking and provides for creation of SRF's or execution from them. During this procedure the execution sequence of the modules is checked to ensure that the required system level setup modules were previously executed as required. In Appendix A of this report an example is provided of interactive execution of the IRSS.

4.3.1.2 Computer Directed Job Setup

Under this procedure the computer is in control. The user is led through a logical sequence of decision steps such that the general characteristics of the desired simulation are defined. Once these general characteristics are identified the user is led through a more detailed sequence which results in identification of the specific modules to be used in setting up a simulation. Finally, the input parameters for each module are requested and a SRF is created. There are two phases to this procedure. In Phase I block diagrams and a question/answer sequence are used in setting up the SMEST. In Phase II a question/answer sequence is used to define the parameters of the modules entered into the SMEST. In Appendix B of this report an example is provided which illustrates the use of the User Aid Processor (UAP).

In Phase I a prototype radar system block diagram is drawn which contains only major subsystems, e.g. antenna, environment, transmitter, receiver, etc. The user is then asked to pick those which he wishes to incorporate into his simulation. Once the user has chosen the subsystems, a refinement subphase is entered wherein a detailed mechanization block diagram is presented for each subsystem. The user is again asked to choose blocks from those available. After the simulation has been defined the user is allowed to insert plots where he desires. Also, a MPF can be defined at this time. The final result of this procedure is a SMEST which is subsequently processed in Phase II.

In Phase II the IFE processor is used to obtain module parameters from the user. Error checking is performed on all user inputs. The output from this phase is a SRF which can be run by using the IRSS command (RUN).

4.3.2 Error Checking and Recovery

An essential component of the modified IRSS is an extensive error checking facility. There are three levels of error checking which are discussed further in the following paragraphs. Whenever an error is detected, recovery procedures are invoked to explain the error to the user and allow him to correct his mistake and recover from the error. Specific examples of error detection are provided in the user's manual.

4.3.2.1 General Syntax Errors

Errors in this category consist of specifying input data in an improper way. This includes alphabetic characters in numeric fields, two decimal points in a field, and failure to put a decimal point in a real field. These errors are relatively easy to correct since they only involve retyping a single line of input.

4.3.2.2 Parameter Value Errors

Errors in this category can result in several ways. One, a parameter value is outside the acceptable range, e.g. specifying the pulse width as a negative number. Two, a combination of parameters produces a condition that cannot be handled, e.g. in FGENXY the combination of SPW, NSUBP, and FS could require more data samples than can be handled. Third, the user types an unrecognized symbolic name in response to a command request or module name request.

4.3.2.3 Module Execution Sequence Errors

Errors in this category result when the user attempts to execute modules in an unmeaningful way. This can result from several conditions. One, failure to initialize a subsystem before using it, e.g. not initializing the clutter model before trying to use it. Two, attempting to process a waveform array that has not been initialized, i.e. empty.

4.3.3 General Information Retrieval

This procedure can be entered by the user at the command level simply by typing "HELP". Once the user has entered this procedure he can retrieve information from the RADSIM data base just by typing the name of the item he would like to know more about.

4.4 CONTROL FLOW

In Figure 4-2 a block diagram is shown which illustrates how program control flow is modified by use of the IRSS command words. In the figure each subsystem of the IRSS is shown. All switches in the figure are shown in the position which they have when the IRSS is started. Next to each switch the IRSS command words are shown which cause the switch to change position. The user is shown on the left as a source of commands and input data and on the right as a sponge to absorb the output data produced by the IRSS. Arrows are used to indicate the data flow direction where it is unidirectional.

At startup the IRSS is in interactive mode with no files open. The user/terminal keyboard is connected directly to the ITS which passes processed information to the WPS. If the user types in a module name the ITS responds by asking for all input data required by the module. When the ITS has collected all required input data it passes this information to the WPS which loads and executes the module requested by the user. If the requested module is a plotter type module then the WPS connects to the GCS to produce a plot of the waveform specified by the user.

To retrieve information from the data base the command HELP is used. When this occurs SW#2 changes position and the user/keyboard is connected to the HELP processor. All subsequent user requests are routed to the HELP processor until DONE is entered which resets SW#2.

To create a RUN file from interactive execution the command OPENR is used. This causes SW#3 to close. All subsequent information entered by the user is stored in the RUN file until a CLOSER command is entered which resets SW#3.

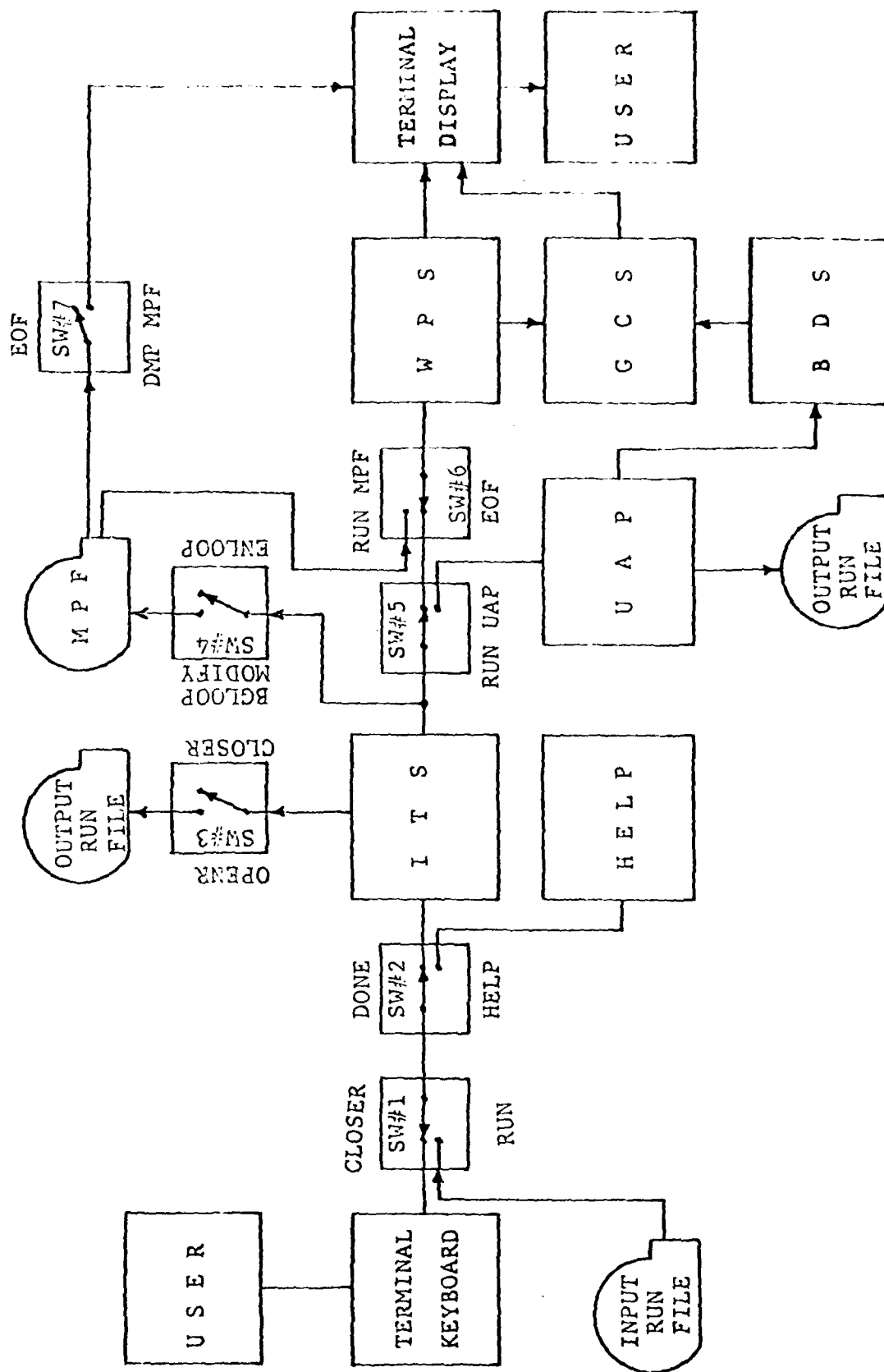


Figure 4-2 Block Diagram of Program Control Flow

To run a simulation from an existing RUN file the command RUN is used. This causes SW#1 to change position and the ITS is connected directly to the input RUN file. The user is required to provide input information only when plots are generated. When a CLOSER command is read from the input RUN file or an EOF is detected then SW#1 is reset to its normal position.

In order to create a Multiple Pass File (MPF) the command BGLOOP is used to close SW#4. All subsequent module execution requests and their associated input parameters are stored in the MPF. The ENLOOP command is used to close the MPF and reset SW#4. If the MPF is to be modified then the MODIFY command is used to close the SW#4 to allow the user to change the parameters of a module. To run a simulation from the MPF the RUN MPF command is used to change the position of SW#6. The WPS now receives all module execution requests and module parameters from the MPF. SW#6 is reset when an EOF is detected in the MPF. The module execution list contained within the MPF can be printed by entering the command DMP MPF. This causes SW#7 to change position to allow data to pass to the display terminal until an EOF is detected in the MPF.

To create a SRF using the User Aid Processor the command RUN UAP is used. This causes SW#5 to change position and the UAP is connected directly to the ITS. The UAP uses the Block Diagram Subsystem (BDS) to create block diagrams via the GCS. The output data from the UAP is placed in a SRF. SW#5 is reset when the user indicates he is finished.

4.5 SUBSYSTEM DESCRIPTIONS

In this subsection the new subsystems which were developed under this effort are discussed. Those subsystems which were not changed during this effort are the GCS, BDS, and the simulation modules which are used in the WPS.

4.5.1 INTERACTIVE TRANSLATOR SUBSYSTEM

The ITS is composed of the Interactive Front End processor and the RADSIM data base. Each of these will be discussed in more detail in the following paragraphs.

4.5.1.1 Interactive Front End (IFE)

The IFE subsystem is designed to serve as an interface between an application program and the user. The IFE has access to a data base which can be used to assist the user. The data base depends on the application program. In Figure 4-3 a functional block diagram is shown of the IFE subsystem attached to an application program. In this figure the application program data base is contained in the IFE file which is disc resident. The IFE multiplexer, upon command from the program, brings various segments of the IFE file into main memory buffer areas. These are shown inside the common area enclosed with dashed lines. The IFE.TIM block represents the IFE modules which interface with the buffer areas. The IFE.UIM block represents the modules which communicate with the user.

An example of some of the capabilities of the IFE are shown in Figure 4-4. This example was generated during the execution of the IRSS module SIMSYS. In the first line the user is asked what procedure he wants to use. The reply is SIMSYS. Next, the user is asked to provide values for the input parameters of the module SIMSYS. The user does not know what to enter for the parameters so he replies with: "?". The IFE then retrieves descriptive information on each parameter and again asks the user to provide a value for each parameter. In this example the user entered values of 11 for N2, 0.2 for FS, 1 for ICFOR, and nothing for NORMFT, RNGCEL, TIME, and ISDUMP.

When the user is asked to enter another command he enters PHENC. Since this module generates a phase code and places it into a waveform file the user must specify where to store it. His reply was PC. The system detected that no waveform named PC had been setup so it allocated a temporary file with that name. Next the user is asked to enter input parameters. The user knows he wants to generate a 13 bit Barker code but can't remember which value of MODEPH is appropriate. Therefore he enters 13 in the field for NSUBP and ? in the field for MODEPH. The IFE responds by printing the valid values of MODEPH and their meanings. The user enters a 1 to cause a Barker code to be generated.

Examples of error detection by the IFE are included in the user manual and will not be reiterated here.

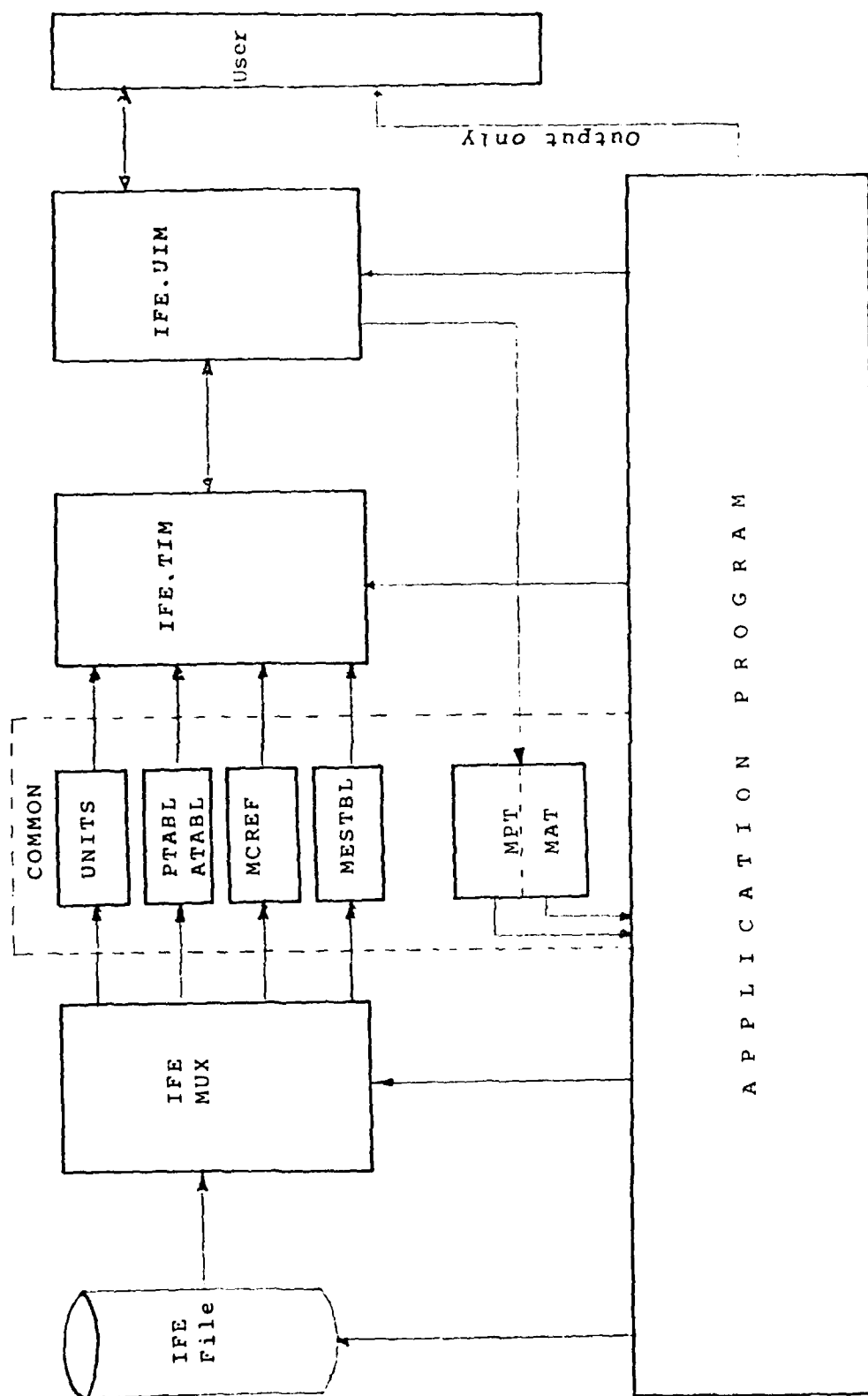


Figure 4-3 IFE Functional Block Diagram

```

Enter CMD
^ SIMSYS

Enter N2      ,FS      ,ICFOR ,NORMFT ,RNGCEL ,
^ 2

Param:N2
Power of 2 used in determining the maximum simulation
waveform length

          9(= N2      (=          17

PARAM:N2      UNITS ARE: No Unit of Measure

ENTER N2
= 11

Param:FS
Simulation sampling rate

          0 0 ( FS

PARAM:FS      UNITS ARE: Ghz

ENTER FS
= 0.2

Param:ICFOR
Fourier Transform flag
0 = Real input data
1 = Complex input data

          0(= ICFOR      (=          1

PARAM:ICFOR   UNITS ARE: Flag

ENTER ICFOR
= 1

Param:NORMFT
Fourier transform normalization flag
0 = Sample increment
1 = ZFFT/DFT by number of samples
2 = ZIFFT by number of samples
3 = 1 and 2 combined
4 = No normalization

          0(= NORMFT      (=          4

PARAM:NORMFT  UNITS ARE: Flag

ENTER NORMFT
=

Param:RNGCEL
Spacing between samples of the clutter impulse response

          TI      (= RNGCEL

PARAM:RNGCEL  UNITS ARE: Nanoseconds

ENTER RNGCEL
=

```

Figure 4-4 Example of IFE Information Retrieval from
RADSIM Data Base


```

Enter TIME      ICODEM ,
^

Simulation Time Span (TSL) = 0 10240: 15

Enter CMD
^ PHENC

Enter 1 Output Waveform Names
= PC

Temporary File Created For PC

Enter NSUP      MODEPH      ICODE      NSR      IPY      ,
^ 13,2

Param:MODEPH
Phase encoder mode flag
#1 Barker code
#2 Pseudo random binary code
#3 User specified code (via ICODE)

      1(= MODEPH )=      3

PARAM:MODEPH UNITS ARE: Flag

ENTER MODEPH
= 1

Enter CMD

```

Figure 4-4 Example of IFE Information Retrieval from
RADSIM Data Base (cont.)

4.5.1.2 RADSIM DATA BASE

The RADSIM data base accessed by the IFE contains the following information:

- Commands
- Module Names
- Module Parameters
- Module Array Elements
- Messages
- Prompt Statements

The module parameter, array element, and name entries include descriptive information such as: units of measure, valid parameter ranges, warnings, and a general description of the item.

4.5.2 HELP PROCESSOR

The HELP processor serves as a direct interactive interface between the user and the RADSIM data base. When the HELP processor is activated all user requests are routed directly to it. In response to each request to define a symbolic name, the HELP processor searches the RADSIM data base for the symbolic name in question. If it is found then all information available pertaining to it is printed. Examples of the HELP processor operation are contained in the user manual and will not be reiterated here.

4.5.3 WAVEFORM PROCESSING SUBSYSTEM

The functional block diagram of the WPS load module is shown in Figure 4-5. This load module is structured so that computer memory requirements are minimized. This is achieved by having all of the simulated waveforms and simulation modules resident in disc files. The waveforms are contained in the waveform files and the simulation modules are resident in an HSTAR file. The GCOS operating system automatically takes care of loading the simulation modules from HSTAR. Therefore, this is not shown in the diagram.

The input commands and data generated by the ITS are the only source of input. These inputs are multiplexed into the SMEST and the Module Parameter Table (MPT)/ Module Array Table (MAT). The SMEST controls the simulation execution and the MPT/MAT contains the module parameters. As each simulation is being performed a MPF can be

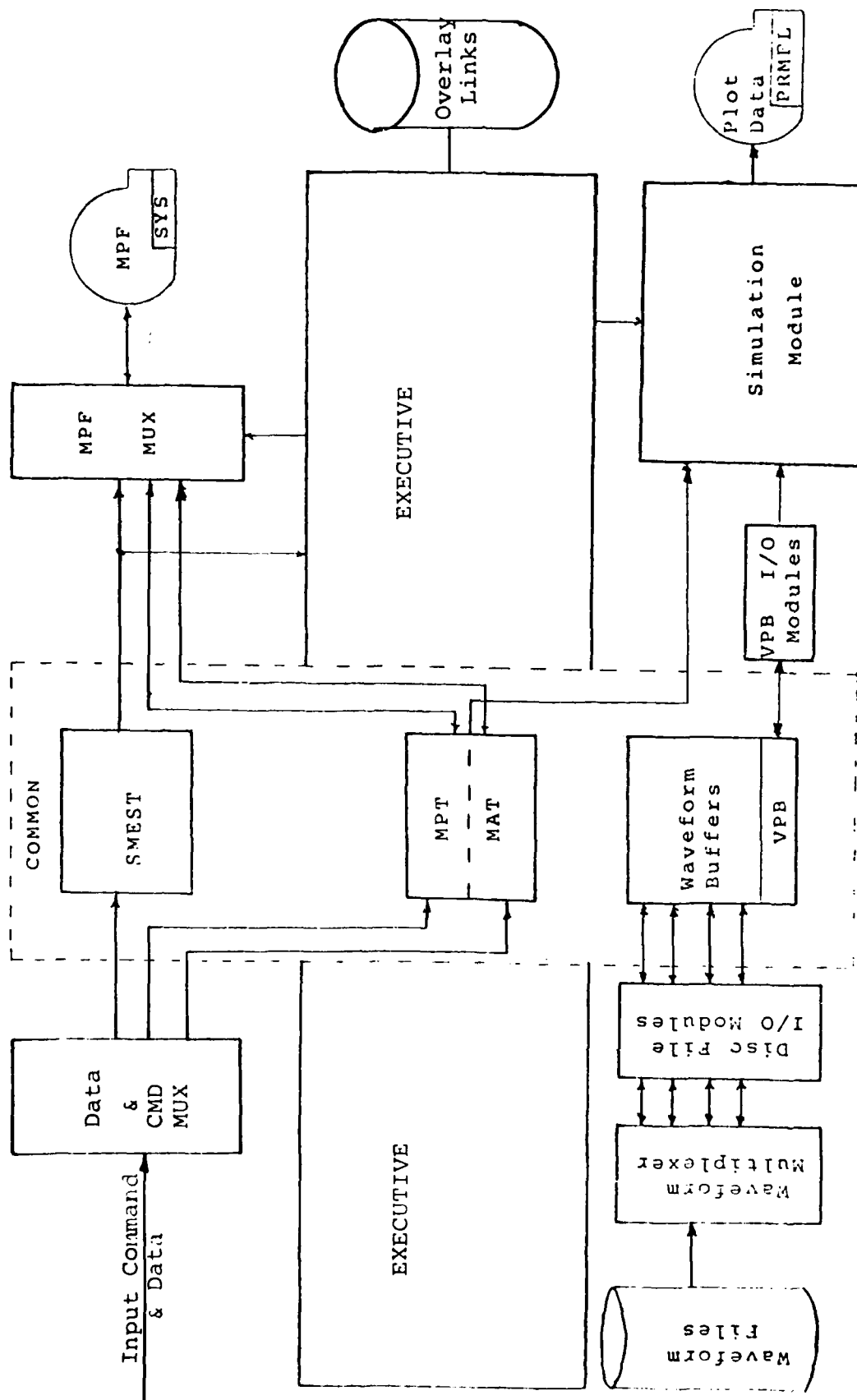


Figure 4-5 WPS Functional Block Diagram

constructed. In the event a simulation is to be modified and rerun, the MPF is used as the primary source of input.

In the figure the Waveform Multiplexer, Disc File I/O modules, Waveform Buffers, and VPB I/O modules perform the manipulations required to move segmented data arrays between memory and the disc files.

4.5.4 USER AID PROCESSOR

The UAP is a simple processor which causes block diagrams of prototype radar subsystems to be drawn and then asks the user which blocks of each subsystem he wishes to include in his simulation. Once the user has answered all the questions regarding the desired simulation structure the IFE processor is used to request the module input parameters for each module. In Appendix B of this report an example is provided which illustrates the use of the UAP in setting up a simulation job. This example includes the block diagrams and the question/answer sequence used.

SECTION 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The results obtained during this investigation have indicated that all objectives of the program have been attained. The primary objective was to modify the IRSS such that it was easier to use. In order to achieve this goal for both the novice and experienced user two different approaches were used in operating the IRSS. First, a totally interactive capability was established such that a user could command modules to be executed and look at the results immediately, thereby getting the fastest possible turnaround. Second, a computer directed procedure was implemented which leads the user through a series of steps in order to set up a simulation. The following is a list of conclusions which relate to the various subsystems that form the IRSS:

1. The incorporation of interactive information retrieval from the IRSS data base has significantly reduced the amount of information that the user must remember and provides for extensive run time error checking. Since the process is interactive the errors are flagged as they occur and the user is required to correct them immediately. The incorporation of the Interactive Front End (IFE) software into the IRSS was the biggest single factor in improving the utility of the IRSS.
2. The User Aid Processor (UAP) was originally developed to aid only the novice user. It now appears that the UAP could be extended to help not only the novice but also the occasional user who knows what he wants to do but has forgotten some of the mechanics of utilization. For the highly experienced user the user directed mode will always be the most efficient.
3. The conversion of the IRSS from batch execution to time sharing was very successful. The CPU time requirements have increased very little due to the added overhead. The memory requirements have dropped from 70K to 34K which is a significant reduction. In performing this conversion the old capabilities of the IRSS were preserved and actually were extended.

4. The DUIS can be used to plot data generated by any TSS program run under GCOS. This is done by using the DUIS plot interface module, PLOTD, which is resident in the H6180 under account CF4IRSS.

To summarize, the IRSS in its current configuration is much more forgiving than its predecessor. The key element in "humanizing" this simulation program was the incorporation of a sophisticated processor to communicate with the user. This processor is capable of detecting user errors as they occur and requesting correction. Also the processor has access to a data base of information on module characteristics, parameters, I/O requirements, etc.

5.2 RECOMMENDATIONS

In view of the results presented in this report and the potential applications of the radar simulation model in evaluating numerous radars, it is recommended that further studies be performed to improve the usefulness of the IRSS. The specific areas that deserve further effort include:

1. Upgrade the capabilities of the DUIS by adding 3 flexible disc drives and 128K bytes of memory. The incorporation of this equipment into the DUIS would provide the following new capabilities:
 - A. 3D plots could be generated which would allow ambiguity diagrams and antenna patterns to be plotted.
 - B. The IRSS data base could be resident in the DUIS thereby reducing the cost of using the H6180.
 - C. A full text editing and word processing capability would be available independent of the H6180.
 - D. The DUIS could function as a standalone computer system which would support source file creation and editing, FORTRAN compiler, load module creation and job execution.
 - E. General block diagrams could be generated for flow charts, viewgraphs, etc.
2. Modify the IRSS to allow more extensive modifications of the Multiple Pass File (MPF) to be made. Examples are deletion, insertion, replacement, and skipping of modules. In addition set up a simple editor for

modifying Simulation RUN Files.

3. Expand the capabilities of the User Aid Processor so that it can be helpful to the occasional user as well as the novice. Provide for prestructured simulations where the simulation structure is laid out for various generic radar systems and the user only has to provide parameters.
4. Expand the data base to provide a description of each module including block diagrams if appropriate.
5. Add run time tests to verify that the user specified input waveforms for a module are compatible with the characteristics of the module. For example, it would be incorrect to specify time domain input waveforms for a module which processes frequency domain data.
6. Add a range and angle track capability, an endoatmospheric chaff model, a more accurate ECM model, and multiple maneuvering target modules.
7. Perform more research into the development of general purpose front end processors for application programs. This would be particularly helpful in supporting the development of some advanced simulations for such systems as the Space Based Radar.
8. Provide for spooling of plot files generated during a simulation such that the IRSS can be run from any terminal and the plots retrieved later via the DUIS.
9. Add a capability for processing a system noise cumulative distribution function (CDF) and noise+target CDF's to generate Pd curves for a system.
10. Incorporate filter design programs into the IRSS so that the user can specify filters in terms of filter type and filter characteristics such as number of poles, corner frequency, rolloff rate, etc. The filter modules currently in the IRSS require that filters be specified in terms of S-plane or Z-plane pole/zero locations or digital filter coefficients.

SECTION 6
REFERENCES

1. Hancock, R. J. and Cleveland, F. H., Endo Atmospheric-Exo Atmospheric Radar Modeling (U), RADC-TR-76-186, Vol. 1, Griffiss AFB, New York, June 76. Part 1 (A030555) Part 2 (A030496) and Part 3 (A030504)
2. Hancock, R. J., Graphics Control System Computer Program Documentation, submitted to RADC, Griffiss AFB, February 79.
3. Hancock, R. J., Interactive Radar Simulator General Description, submitted to RADC, Griffiss AFB, May 79.

A P P E N D I X A
A N E X A M P L E O F
I N T E R A C T I V E E X E C U T I O N

In this appendix an example is provided which illustrates the use of the IRSS in an interactive manner. In this example a Simulation RUN File (SRF) and a Multiple Pass File (MPF) were created.

In the listing which follows, copies of the plots that were generated have been inserted in the text at the point of occurrence. It should be noted that after each plot was generated the statement "Type CR When Ready" was printed. If a hardcopy of the plot was desired it was made at that time. In the listing all user replies are underlined. Null responses are indicated by an underline only.

In the example which follows a 13 bit Barker code was generated and passed through a filter. Then noise was added to it and the resulting waveform was digitized and passed through a digital decoder. It should be noted that this example was chosen only to illustrate the use of the IRSS in an interactive manner and is not intended to represent any particular system design. In this example the use of the SRF and the MPF is not shown.

YFORT

* RUN IRSS

IRSS Started

For a List of Commands Type: ? at CMD Level

Enter CMD

^ OPENR

Enter RERUN File Name

= DIGDECOD

File Does not Exist-DIGDECOD

Creating New File

Enter CMD

^ SIMSYS

Enter N2 ,FS ,ICFOR ,NORMFT ,RNGCEL ,

^ 11,2

Enter TIME ,ISDUMP ,

Simulation Time Span (TTL) = 0.10240E 05

Enter CMD

^ PHENC

Enter 1 Output Waveform Names

= PC

Temporary File Created For:PC

Enter NSUBP ,MODEPH ,ICODE ,NSR ,IPY ,

^ 13,1

Enter CMD

^ BGLOOP

Enter TIME ,NREPET ,

^ 0

Enter CMD
 ^ PCODXY
 Enter 1 Input Waveform Names
 = PC
 Enter 2 Output Waveform Names
 = A1,A2
 Temporary File Created For:A1
 Temporary File Created For:A2
 Enter IPHM ,SIMF0 ,NORMFT ,SPW ,NSUBP ,
 ^ ,,,50.
 Enter SWTIM ,RISTIM ,FALTIM ,TSTART ,VPEAK ,
 ^ 10.,10.,10.,75.

Enter CMD
 ^ XFRM
 Enter 2 Input Waveform Names
 = A1,A2
 Enter 2 Output Waveform Names
 = B1,B2
 Temporary File Created For:B1
 Temporary File Created For:B2

Enter CMD
 ^ PLOTR
 Enter 1 Input Waveform Names
 = A1
 Enter ST ,RNG ,NSKP ,
 ^ 0.,1000.
 Plot Xfer Started
 Xfer Complete
 = 0

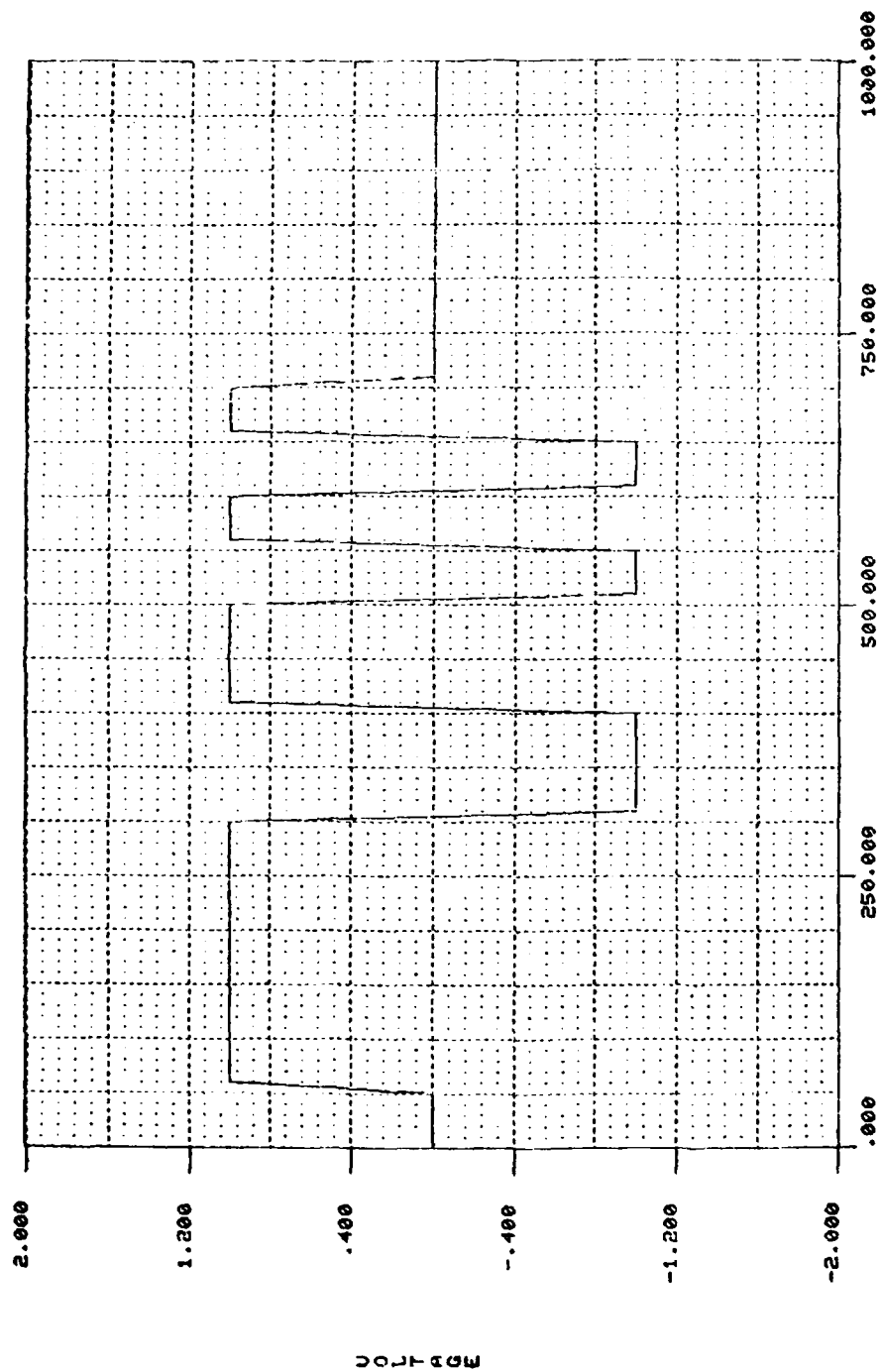


Figure A-1 Phase Coded Waveform

Type CR When Ready
=

Enter CMD

^ PLOTDB

Enter 2 Input Waveform Names
= B1,B2

Enter ST ,RNG ,NSKP ,

Plot Xfer Started
Xfer Complete
= 0

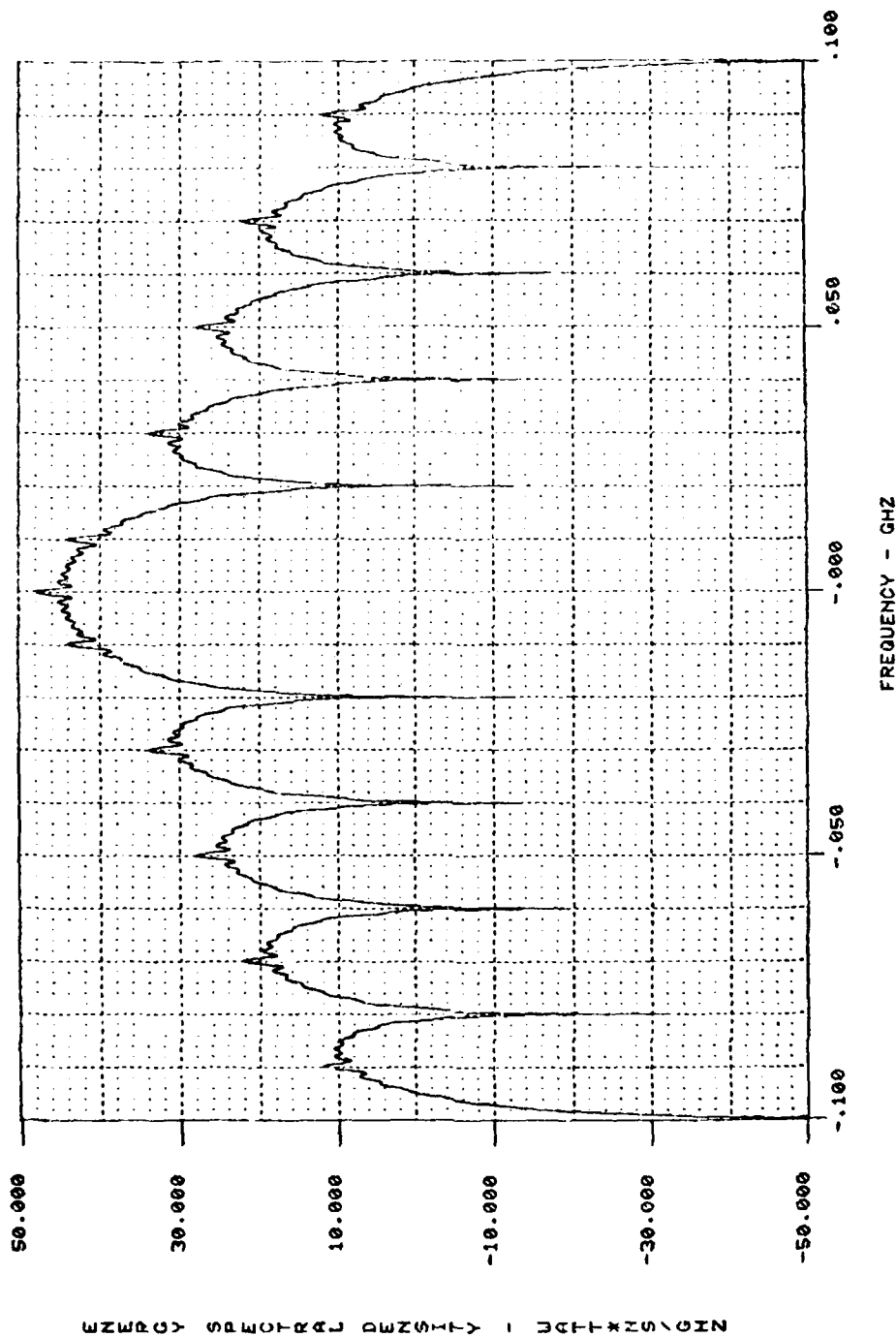


Figure A-2 Spectrum of Phase Coded Waveform

Type CR When Ready

Enter CMD

^ FILT

Enter 2 Input/Output Waveform Names

= B1,B2

Enter SF

^ 1.0E-08

Enter NP

^ 4

ENTER FPOLE (2, 4)

^ -0.0092388,0.0038268

= -0.0092388,-0.0038268

= -0.0038268,0.0092388

= -0.0038268,-0.0092388

Enter NZ

^ 0

Enter CMD

^ IXFRM

Enter 2 Input Waveform Names

= B1,B2

Enter 2 Output Waveform Names

= A1,A2

Enter CMD
^ NOISE
Enter 2 Input/Output Waveform Names
= A1,A2

Enter SIGMA ,
^ .001

Enter CMD
^ PLOTTR
Enter 1 Input Waveform Names
= A1

Enter ST ,RNG ,NSKP ,
^ ,1000.

Plot Xfer Started
Xfer Complete
= 0

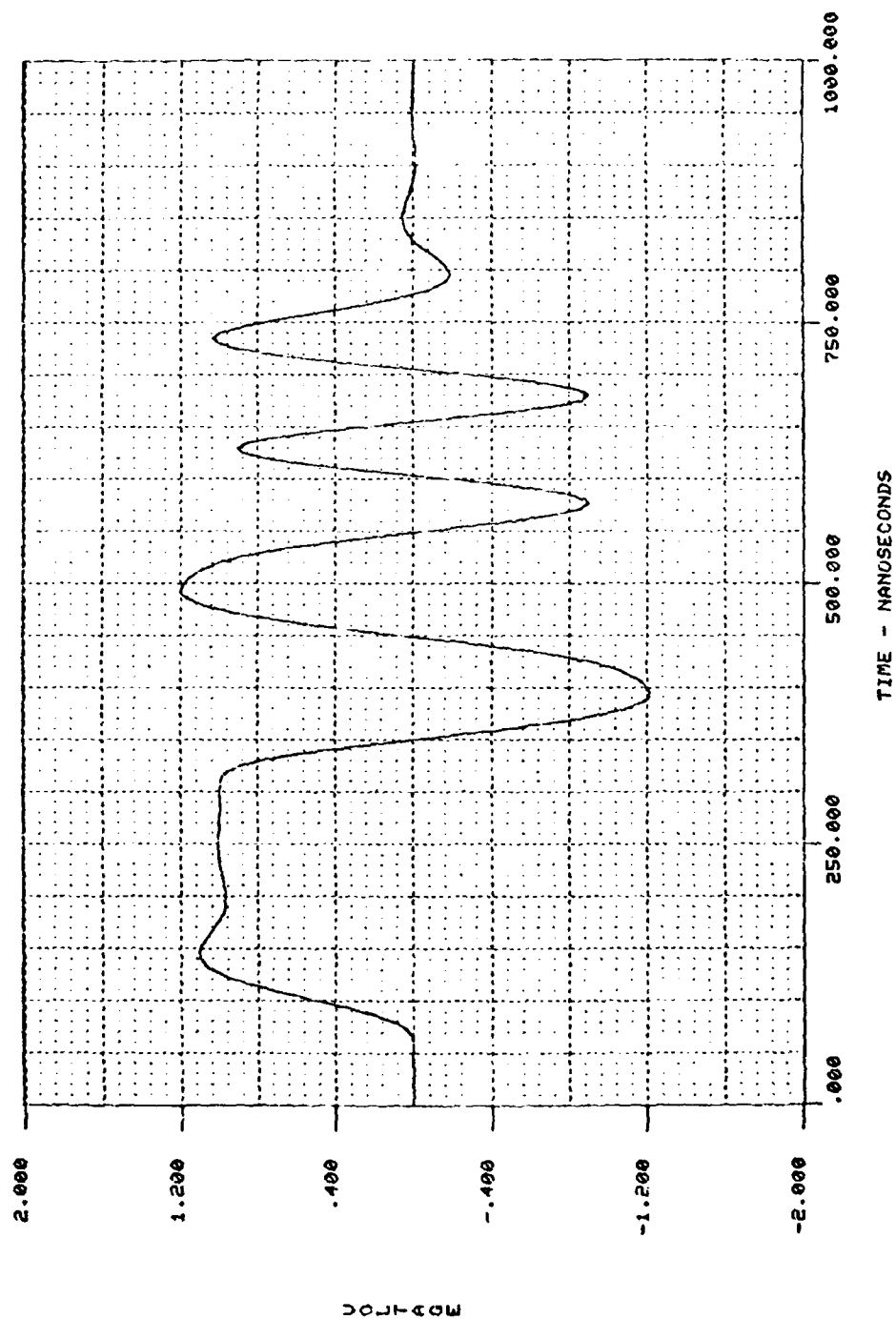


Figure A-3 Phase Coded Waveform + Noise
(Noise SIGMA = 0.001)

Type CR When Ready

=

Enter CMD

^ PLOTTR

Enter 1 Input Waveform Names

= A2

Enter ST ,RNG ,NSKP

^ ,1000.

Plot Xfer Started

Xfer Complete

= 0

Type CR When Ready

=

Enter CMD

^ ADC

WARNING: Nonlinear Transfer functions
may generate harmonics which exceed FS/2

Enter 1 Input Waveform Names

= A1

Enter 1 Output Waveform Names

= I1

Temporary File Created For: I1

Enter XLSB ,NBITS ,IROFF ,ADCFS ,

^ .05,10,,.020

Enter CMD

^ PLOTTR

Enter 1 Input Waveform Names

= I1

Enter ST ,RNG ,NSKP ,

^ ,1000.

Plot Xfer Started

Xfer Complete

= 0

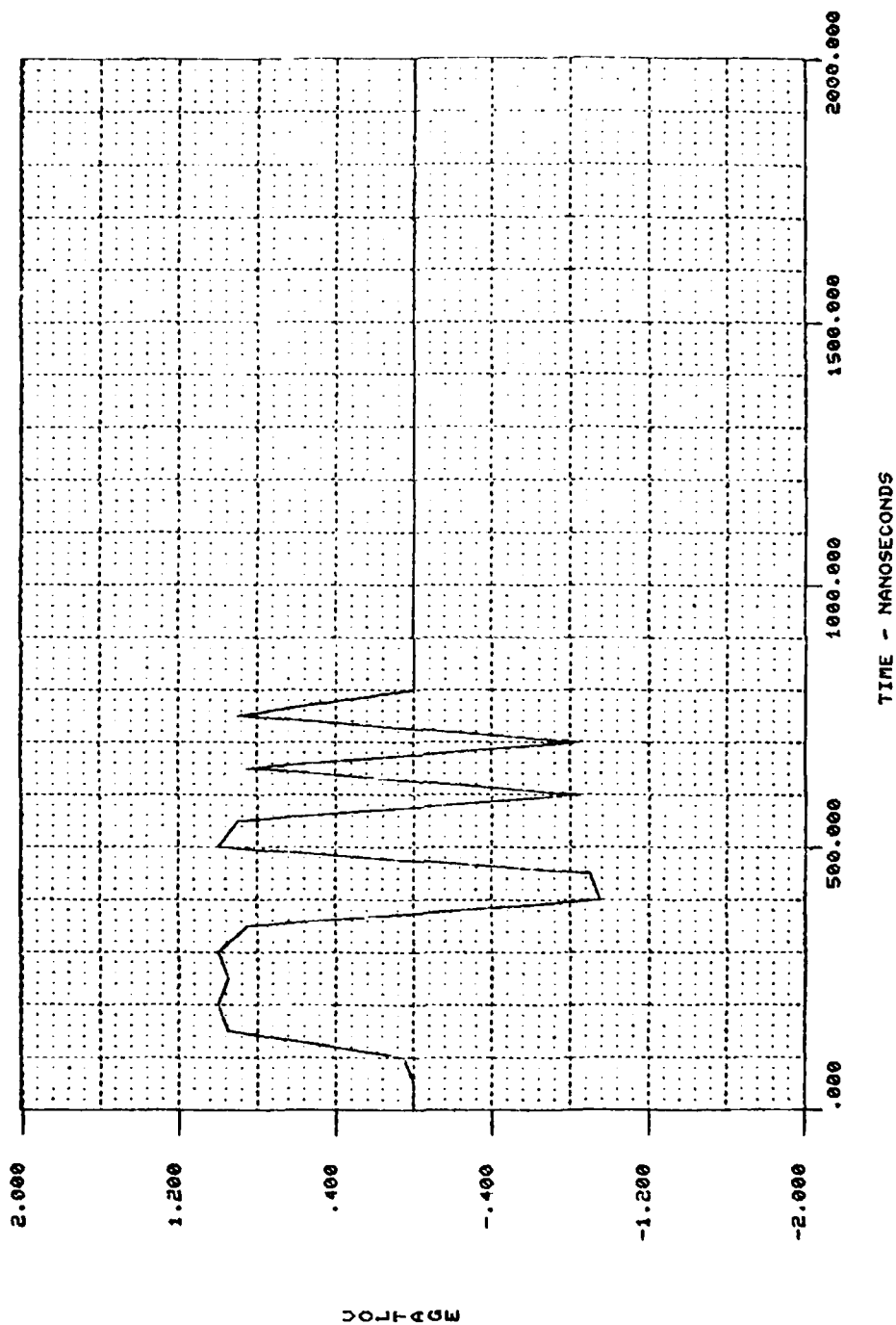


Figure A-4 Analog to Digital Converter Output
(Noise SIGMA = 0.001)

Type CR When Ready

Enter CMD

^ IHLIM

WARNING: Nonlinear Transfer functions
may generate harmonics which exceed FS/2

Enter 1 Input Waveform Names

= I1

Enter 1 Output Waveform Names

= I1

Enter CMD

^ PLOTTR

Enter 1 Input Waveform Names

= I1

Enter ST ,RNG ,NSKP ,

^ ,1000.

Plot Xfer Started
Xfer Complete

= 0

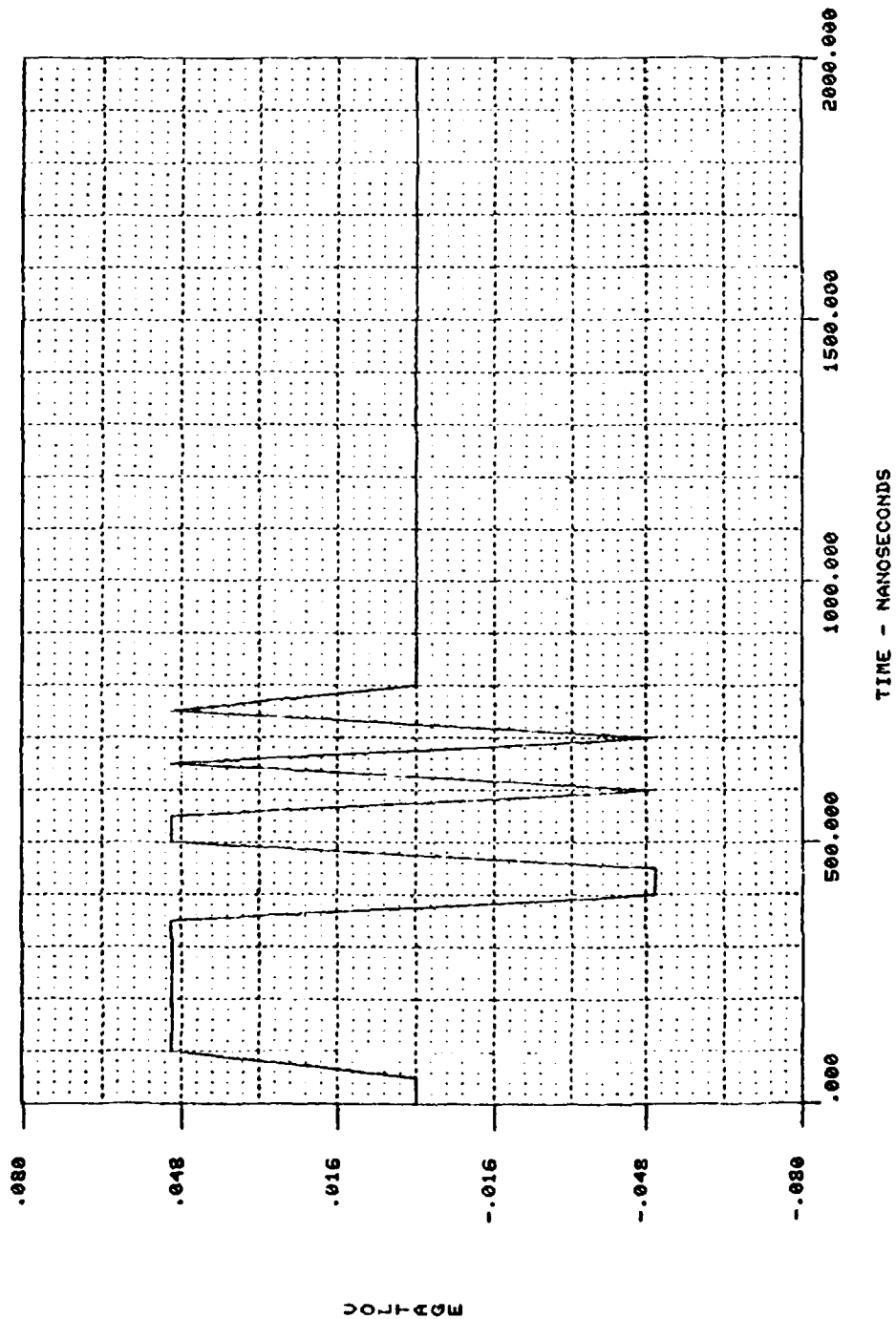


Figure A-5 Hard Limiter Output
(Noise SIGMA = 0.001)

Type CR When Ready

=

Enter CMD

^ DIGTFL

WARNING: Nonlinear Transfer functions
may generate harmonics which exceed FS/2

Enter 1 Input Waveform Names

= I1

Enter 1 Output Waveform Names

= I2

Temporary File Created For: I2

Enter ISPAC ,

^ 1

Enter NTAPS ,

^ 13

ENTER ITAP (2, 13)

^ 1,1

= -1,1

= 1,1

= -1,1

= 1,1

= 1,1

= -1,1

= -1,1

= 1,1

= 1,1

= 1,1

= 1,1

= 1,1

Enter CMD

^ PLOTTR

Enter 1 Input Waveform Names

= I2

Enter ST ,RNG ,NSKP ,

^ ,1000.

Plot Xfer Started
Xfer Complete

= 0

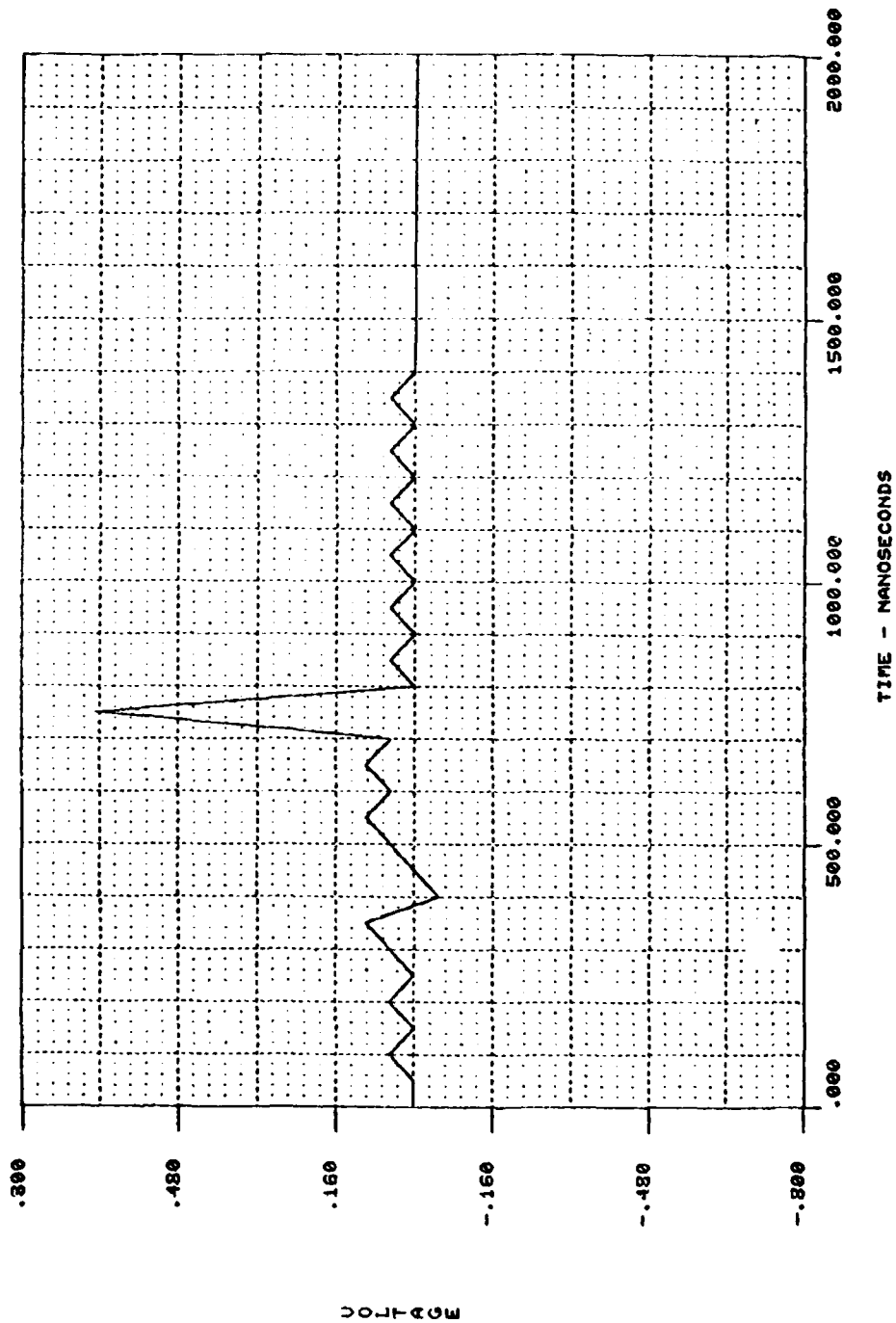


Figure A-6 Digital Decoder Output
(Noise SIGMA = 0.001)

Type CR When Ready

Enter CMD

^ ENLOOP
Enter CMD

^ CLOSER
Enter CMD

^ STOP
IRSS Terminating

* BYE

The simulation that was setup in this example was subsequently rerun with the noise $\text{SIGMA} = 0.5$. The plots obtained for this case are shown in Figures A-7, -8, -9, and -10.

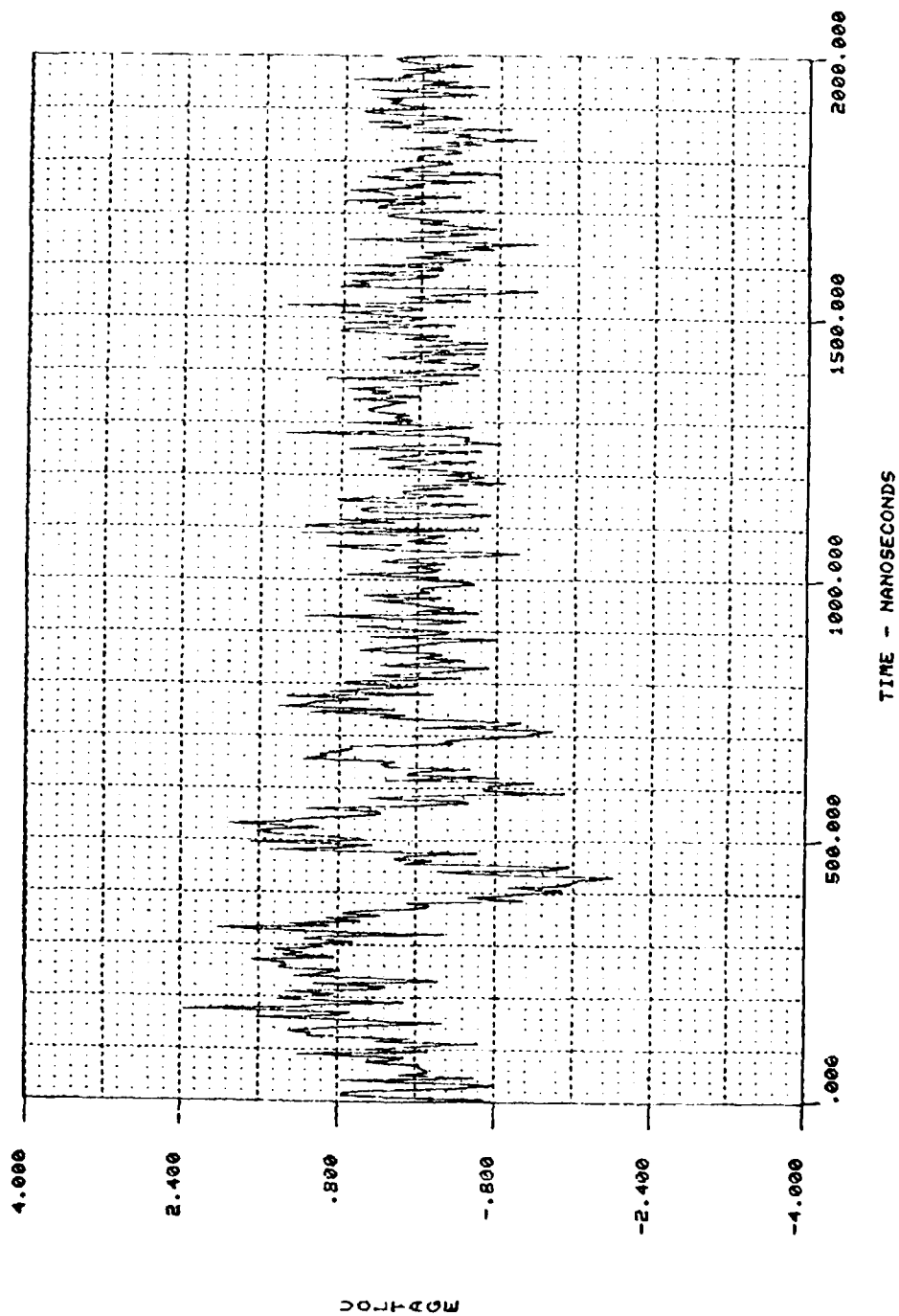


Figure A-7 Phase Coded Waveform + Noise
(Noise SIGMA = 0.5)

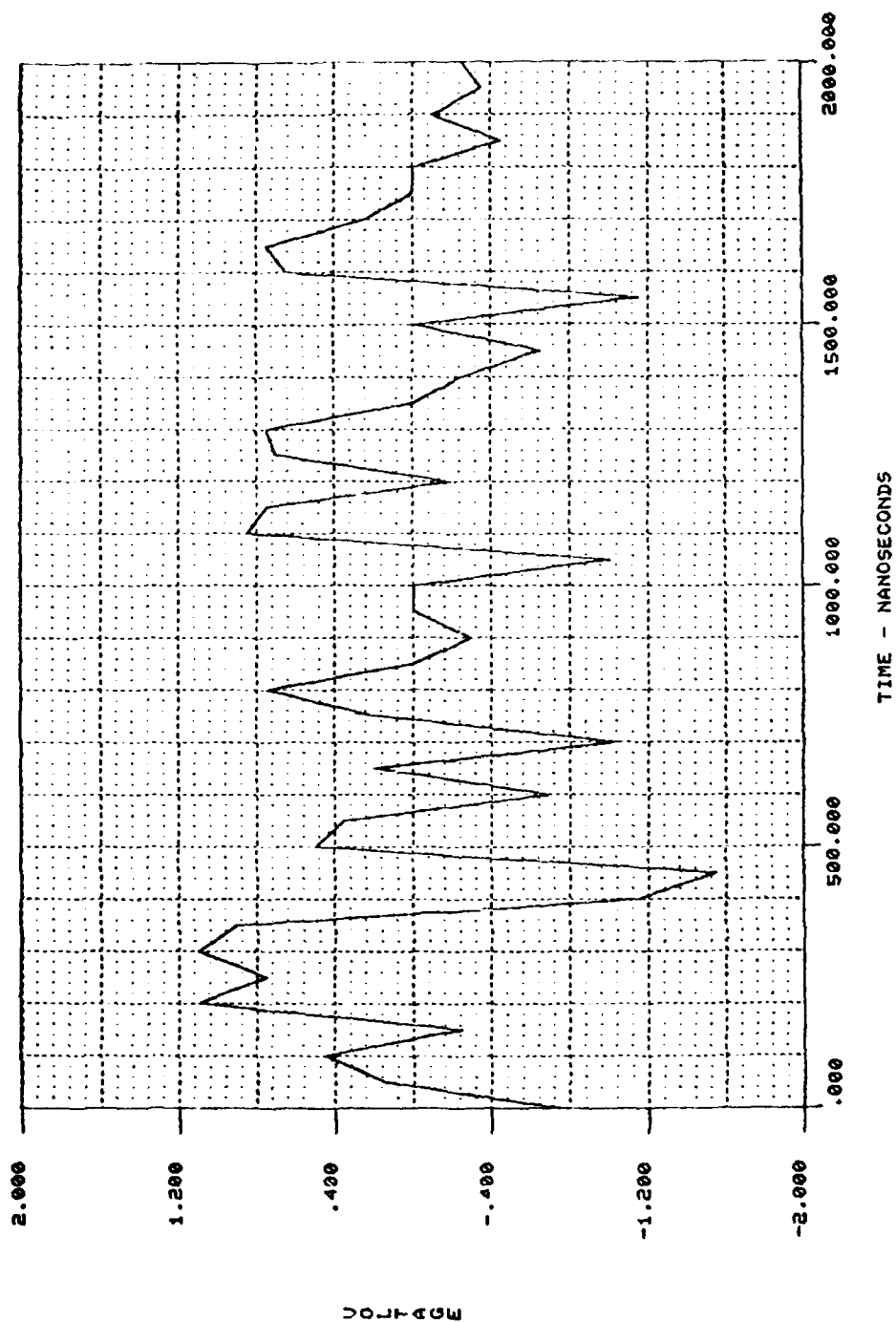


Figure A-8 Analog to Digital Converter Output
(Noise SIGMA = 0.5)

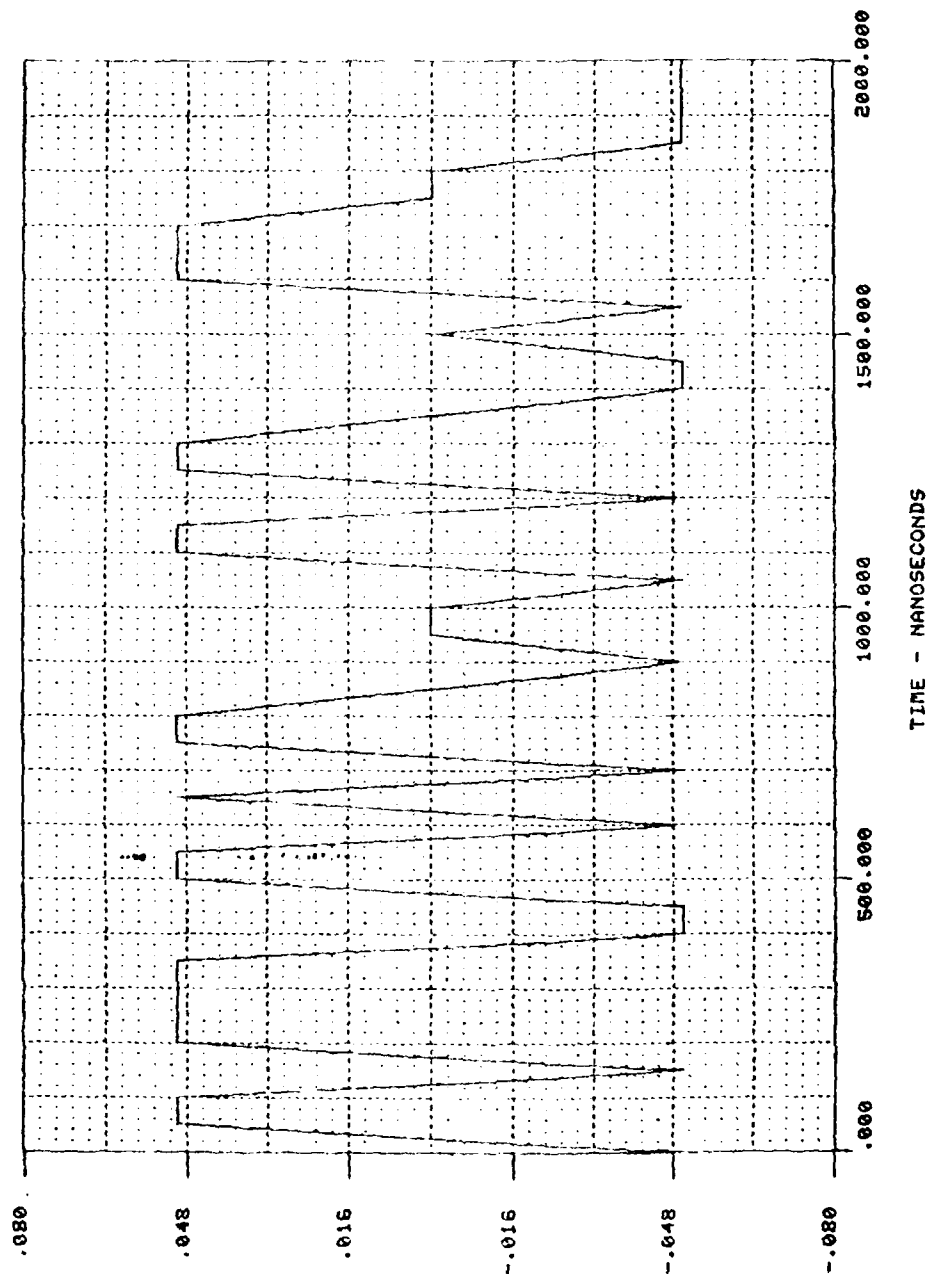


Figure A-9 Hard Limiter Output
(Noise SIGMA = 0.5)

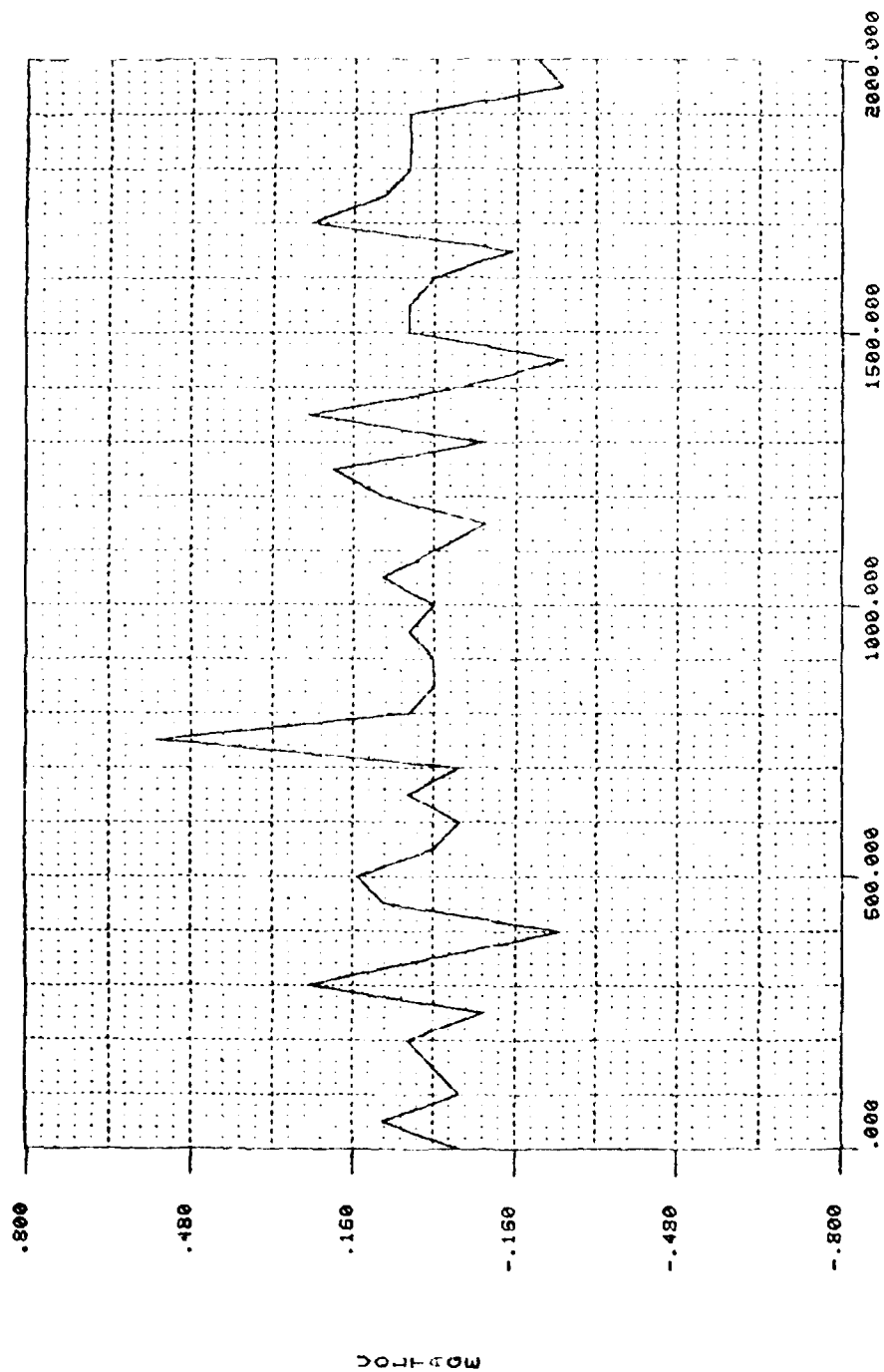


Figure A-10 Digital Decoder Output
(Noise SIGMA = 0.5)

A P P E N D I X B
A N E X A M P L E O F
U S E R A I D P R O C E S S O R E X E C U T I O N

In this appendix an example is provided which illustrates the use of the User Aid Processor (UAP) in setting up a simulation. The UAP utilizes block diagrams to aid the user in visualizing the structure of the available options. These block diagrams are shown in Figures B-1 through B-6. In Figure B-1 the available subsystems are shown. In Figures B-2 through B-6 the prototype block diagrams of the subsystems are shown.

In the listing which follows the occurrence of each block diagram is indicated by its Figure number and all user replies are underlined. Null responses are indicated by an underline with no characters above it.

In the example which follows a 50 bit Pseudo Random Binary Sequence (PRBS) phase coded waveform was generated, then contaminated with noise and filtered. The resulting waveform was then passed through a matched filter and a CFAR processor. It should be noted that this example was chosen only to illustrate the use of the UAP and is not intended to represent any particular system design. Execution of the simulation RUN File (PRBSCODE) created in this example is shown in Appendix C.

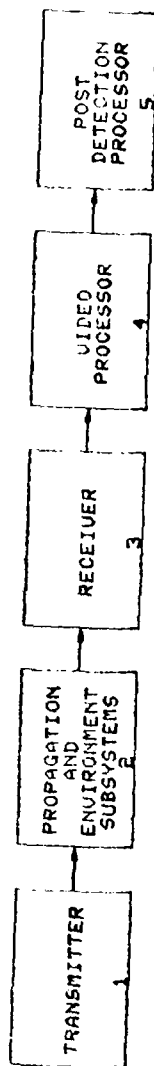


Figure B-1 RADAR SYSTEM PROTOTYPE BLOCK DIAGRAM

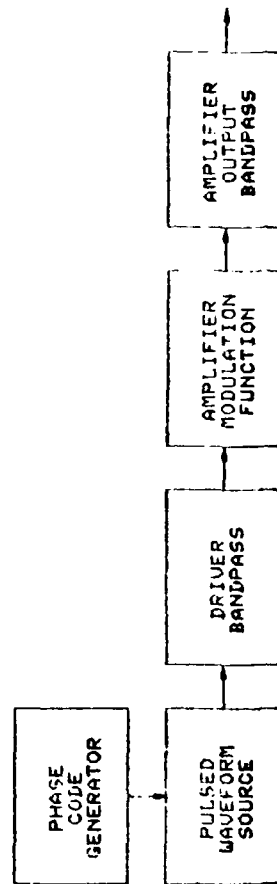


Figure B-2 TRANSMITTER BLOCK DIAGRAM
(Subsystem Index = 1)

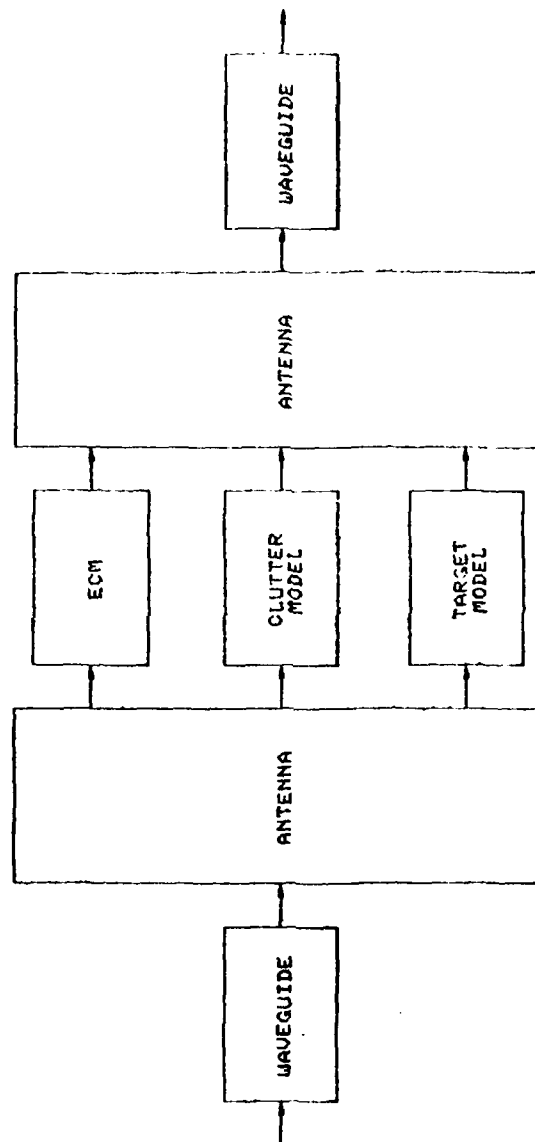


Figure B-3 PROPAGATION AND ENVIRONMENT BLOCK DIAGRAM
(Subsystem Index = 2)

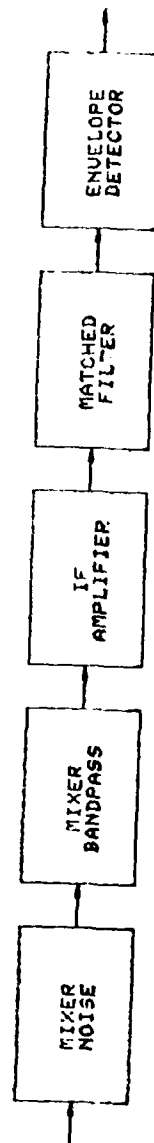


Figure B-4 RECEIVER BLOCK DIAGRAM
(Subsystem Index = 3)

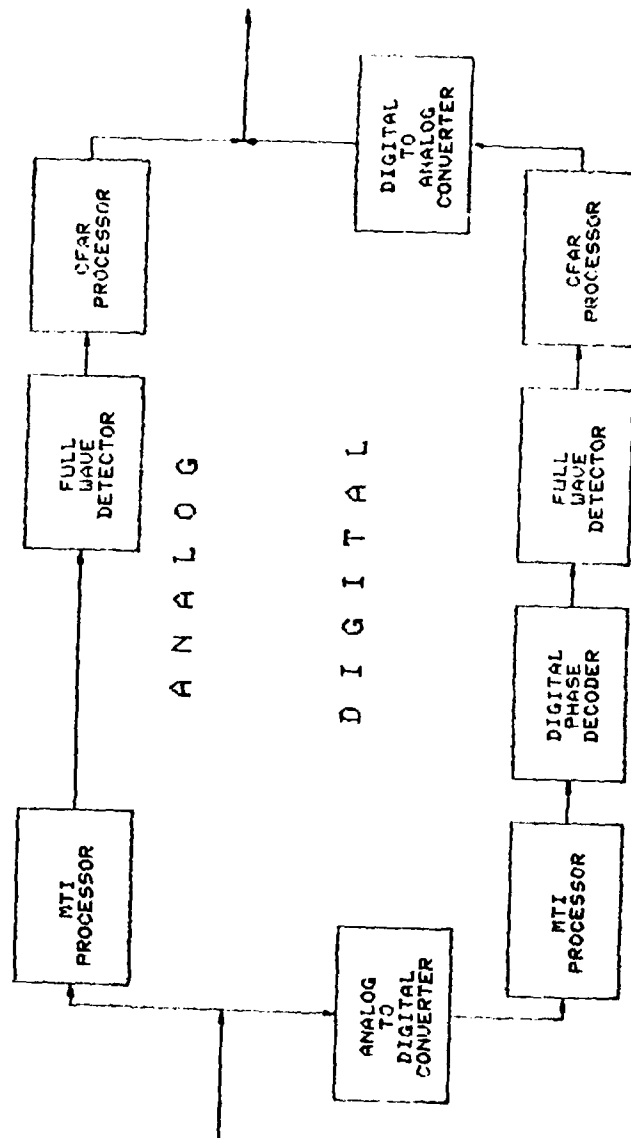


Figure B-5 VIDEO PROCESSING BLOCK DIAGRAM
(Subsystem Index = 4)

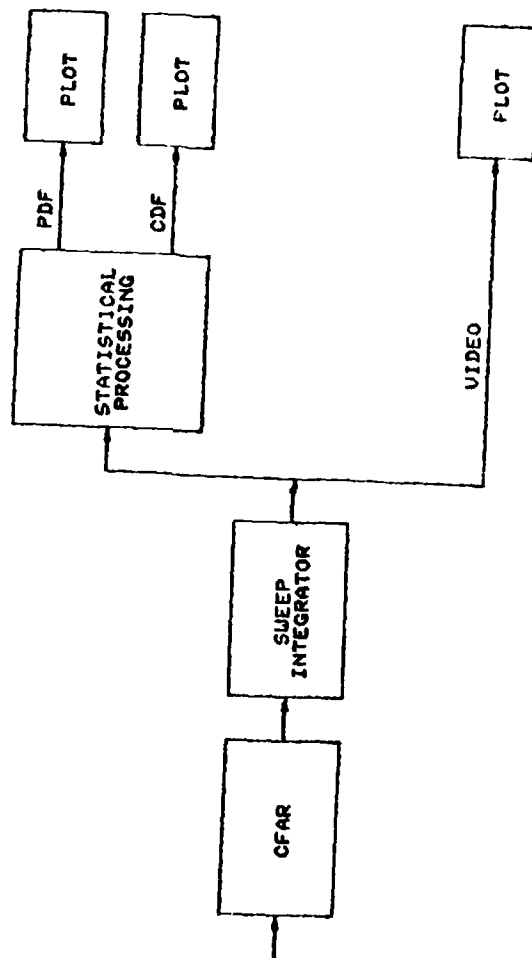


Figure B-6 POST DETECTION PROCESSING BLOCK DIAGRAM
(Subsystem Index = 5)

* YFORT

* RUN IRSS

IRSS Started

For a List of Commands Type: ? at CMD Level

Enter CMD

^ RUN UAP

Block Diag Xfer Started

= 0

Figure B-1

Enter Indices of Subsystems to be Set Up:

1 Transmitter

2 Propagation and Envr.

3 Receiver

4 Video Processing

5 Post Detection Processing

= 1,3,4,5

Block Diag Xfer Started

= 0

Figure B-3

Phase Coded Waveform?

= Y

Driver Bandpass Included?

= N

Pulse Shape Options Follow:

1 A Standard Weighting Function

2 User Specified Complex Envelope

3 None

Enter INDEX

= 3

Amplifier Bandpass Included?

= N

Block Diag Xfer Started

= 0

Figure B-4

Include Mixer Noise?

= Y

Mixer Bandpass?

= Y

Amplifier Gain?

= N

Amplifier Bandpass?

= N

Available Hatched Filter Options Follow:

- 1 Xmit Waveform Conjugate
- 2 S-Plane Specification
- 3 Direct Specification
- 4 Transversal Filter (SAW Device)
- 5 None

Enter Index

= 1

Detector?

= Y

Block Diag Xfer Started

= 0

Digital or Analog? (D or A)

= A

MTI Processor?

= N

CFAR Processor?

= Y

Block Diag Xfer Started

= 0

CFAR Processor?

= N

Sweep Integrator?

= N

Statistical Analysis?

= N

Plot Video?

= Y

Figure B-5

Figure B-6

MODULE EXECUTION SEQUENCE TABLE

MSN	MODULE	WAVEFORM LIST
1	PHENC	PC
2	PCODXY	PC A1 A2
3	XFRM	A1 A2 B1 B2
4	MATFIL	B1 B2 X1 X2
5	NOISE	A1 A2
6	XFRM	A1 A2 B1 B2
7	FILT	B1 B2
8	MOVE	X1 X2 A1 A2
9	MULT	A1 A2 B1 B2
10	IXFRM	B1 B2 A1 A2
11	XYTOM	A1 A2 A1
12	CLEART	A2
13	XFRM	A1 A2 B1 B2
14	FWDET	A1
15	CFAR	A1 A1
16	PLOTTR	A1

Insert A Plot?

= Y

Enter Module Sequence Number (MSN)

= 2

Enter Plot Type Index:

- 1 Single Channel
- 2 Envelope
- 3 Envelope Squared
- 4 Envelope in dB

= 1

Enter Waveform to be Plotted

= A1

Insert A Plot?

= Y

Enter Module Sequence Number (MSN)

= 3

Enter Plot Type Index:

- 1 Single Channel
- 2 Envelope

= 4

- 3 Envelope Squared
- 4 Envelope in dB

= 4

Enter Waveforms to be Plotted:

Waveform #1 =

= B1

Waveform #2 =

= B2

Insert A Plot?
 = Y
 Enter Module Sequence Number (MSN)
 = 5
 Enter Plot Type Index:
 1 Single Channel
 2 Envelope
 3 Envelope Squared
 4 Envelope in dB
 = 1
 Enter Waveform to be Plotted
 = A1
 Insert A Plot?
 = Y
 Enter Module Sequence Number (MSN)
 = 11
 Enter Plot Type Index:
 1 Single Channel
 2 Envelope
 3 Envelope Squared
 4 Envelope in dB
 = 1
 Enter Waveform to be Plotted
 = A1
 Insert A Plot?
 = N
 Set Up Multiple Pass Loop?
 = Y
 Enter First Module Sequence Number of Loop
 = 2
 Enter Last Module Sequence Number of Loop
 = 16
 Enter RERUN File Name
 = PRBSCODE
 File Does not Exist-PRBSCODE
 Creating New File

At this point in the setup procedure the dimensions
 (Frequency X Time) of the sample space to be used
 in the simulation must be specified. Since this
 aspect of the simulation setup is of critical importance
 more information is available if you need it. Do
 you need more information?
 = Y

1. The frequency dimension is determined by the sampling rate parameter, FS
2. The time dimension is determined by the maximum number of samples to be used in representing the waveforms. For compatibility with the Discrete Fourier Transform algorithm used in the IRSS, the maximum number of samples must be a power of 2. Therefore, the parameter N2 is used to determine the time dimension of the sample space (TTL) as follows:

$$TTL = (1.0/FS) * (2 * N2)$$

Do you need more information to select FS and N2?

= Y

1. Normally the spectral width of the transmitted waveform will determine the minimum acceptable value of FS. If FS is too small, then frequency components greater than FS/2 will be folded into the sample space (aliased) and cause errors.
2. To select a minimum value of TTL you must roughly estimate the total time span required to represent the transmitted waveform after it has been convolved with all transfer functions included in the simulation. Once you estimate the minimum TTL, multiply it by FS to estimate the number of samples required. Then round that number up to the nearest power of 2. The following is provided for your convenience:

N2	2*N2
9	512
10	1024
11	2048
12	4096
13	8192
14	16384

If the number of samples chosen to represent the waveforms is inadequate, time wraparound will occur.

SIMSYS

Enter N2 ,FS ,ICFOR ,NORMFT ,RNGCEL ,
 ^ 11,2

Enter TIME ,ISDUMP ,

PHENC

Enter NSUBP ,MODEPH ,ICODE ,NSR ,IPY ,
 ^ 50,2,,8

BCLOOP

Enter TIME ,NREPET ,

^ 0

PCODXY

Enter IPHM ,SIMFO ,NORMFT ,SPW ,NSUBP ,

^ ,,,50.

Enter SWTIM ,RISTIM ,FALTIM ,TSTART ,VPEAK ,

^ ,,,500.

PLOTTT

XFRM

PLOTDB

MATFIL

NOISE

Enter SIGMA ,

^ 1.

PLOTTT

XFRM

FILT

Enter SF ,

^ .036

Enter NP ,

^ 1

ENTER FPOLE (2, 1)

^ -0.03,0.0

Enter NZ ,

^ 0

MOVE

MULT

IXFRM

XYTOM

PLOTTT

CLEART

XFRM

FWDET

CFAR

Enter TAUG , TMIDL ,

?

Param:TAUG

Width of averaging window used in determining video gain

PARAM:TAUG UNITS ARE: Nanoseconds

ENTER TAUG

= 1000.

Param:TMIDL

Width of the central region not included in averaging window

TI <= TMIDL

PARAM:TMIDL UNITS ARE: Nanoseconds

ENTER TMIDL

= 100.0

PLOTT

ENLOOP

Create Another Run File?

= N

Enter CMD

APPENDIX C
AN EXAMPLE OF
SIMULATION RUN FILE EXECUTION

In this appendix an example is provided which illustrates the execution of a simulation from input data contained in a Simulation RUN File (SRF). The SRF used in this example (PRBSCODE) was generated through the use of the UAP. Creation of the RUN file PRBSCODE is documented in Appendix B.

In the listing which follows copies of the plots produced by the IRSS have been inserted in the text at the point of occurrence. It should be noted that after each plot was generated the statement "Type CR When Ready" was printed. This caused execution of the simulation to cease in order that a hardcopy of the plot could be made on the conversation monitor. In the listing all user replies are underlined. Null responses are indicated by an underline only.

RUN IRSS

IRSS Started
For a List of Commands Type: ? at CMD Level
Enter CMD

^ RUN PRBSCODE

Enter CMD

SIMSYS
N2 = 11,FS = 2.0000E-01,
\$
Simulation Time Span (TTL) = 0.10240E 05

Enter CMD

PHENC
Enter 1 Output Waveform Names
PC
Temporary File Created For:PC
NSUBP = 50,MODEPH= 2,NSR =
\$

Enter CMD

BGLOOP
NREPET= 0,
\$

Enter CMD

PCODXY
Enter 1 Input Waveform Names
PC
Enter 2 Output Waveform Names
A1 A2
Temporary File Created For:A1
Temporary File Created For:A2
SPW = 5.0000E 01,TSTART= 5.0000E 02,
\$

Enter CMD

PLOTTR
Enter 1 Input Waveform Names
A1

Enter ST ,RNG ,NSKP ,
A 0.,4000.

Plot Xfer Started
Xfer Complete
= 0

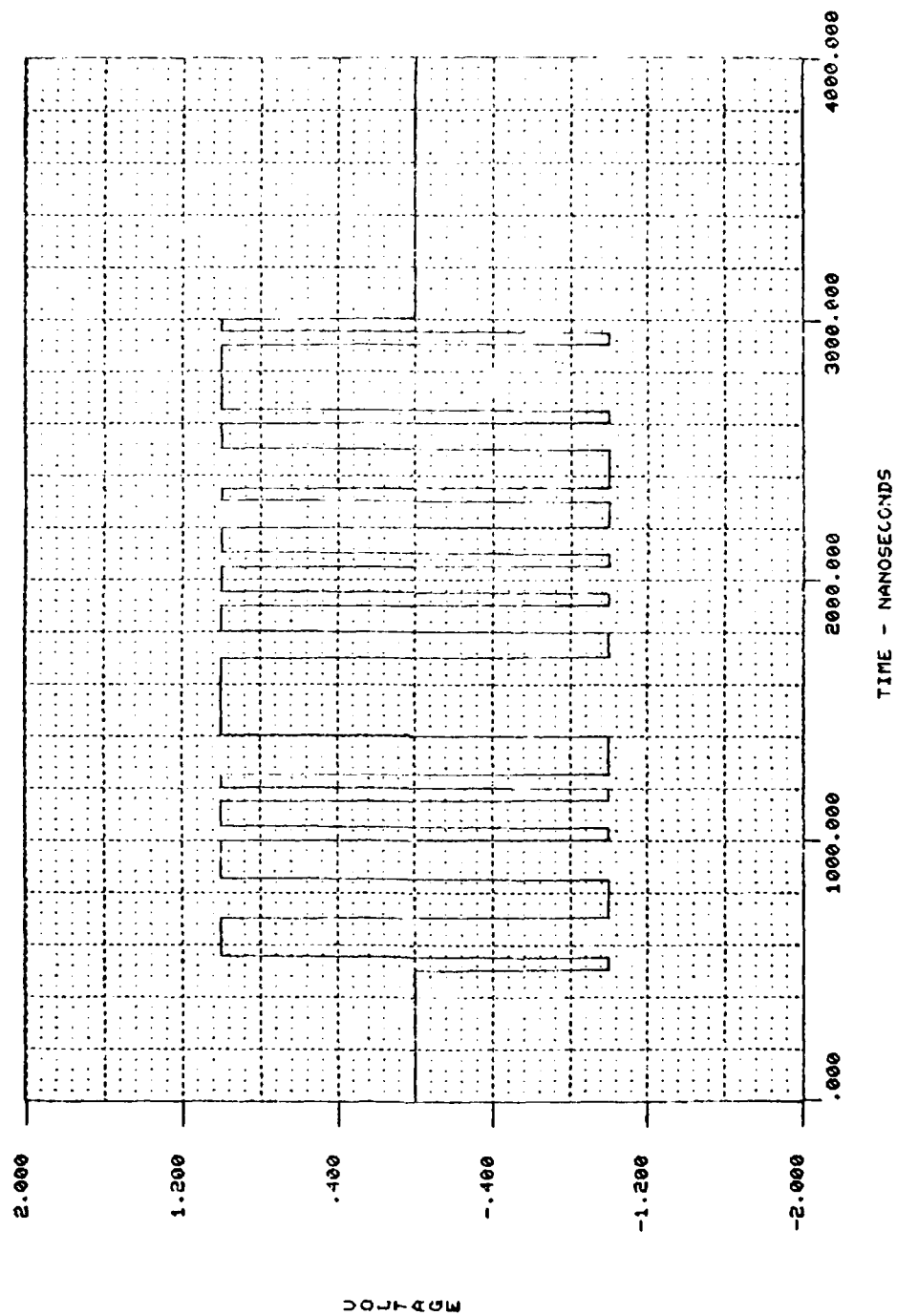


Figure C-1 PRBS Waveform

Type CR When Ready

=

Enter CMD

XFRM

Enter 2 Input Waveform Names

A1 A2

Enter 2 Output Waveform Names

B1 B2

Temporary File Created For:B1

Temporary File Created For:B2

Enter CMD

PLOTDB

Enter 2 Input Waveform Names

B1 B2

Enter ST ,RNG ,NSKP ,

^ , ,2

Plot Xfer Started

Xfer Complete

= 0

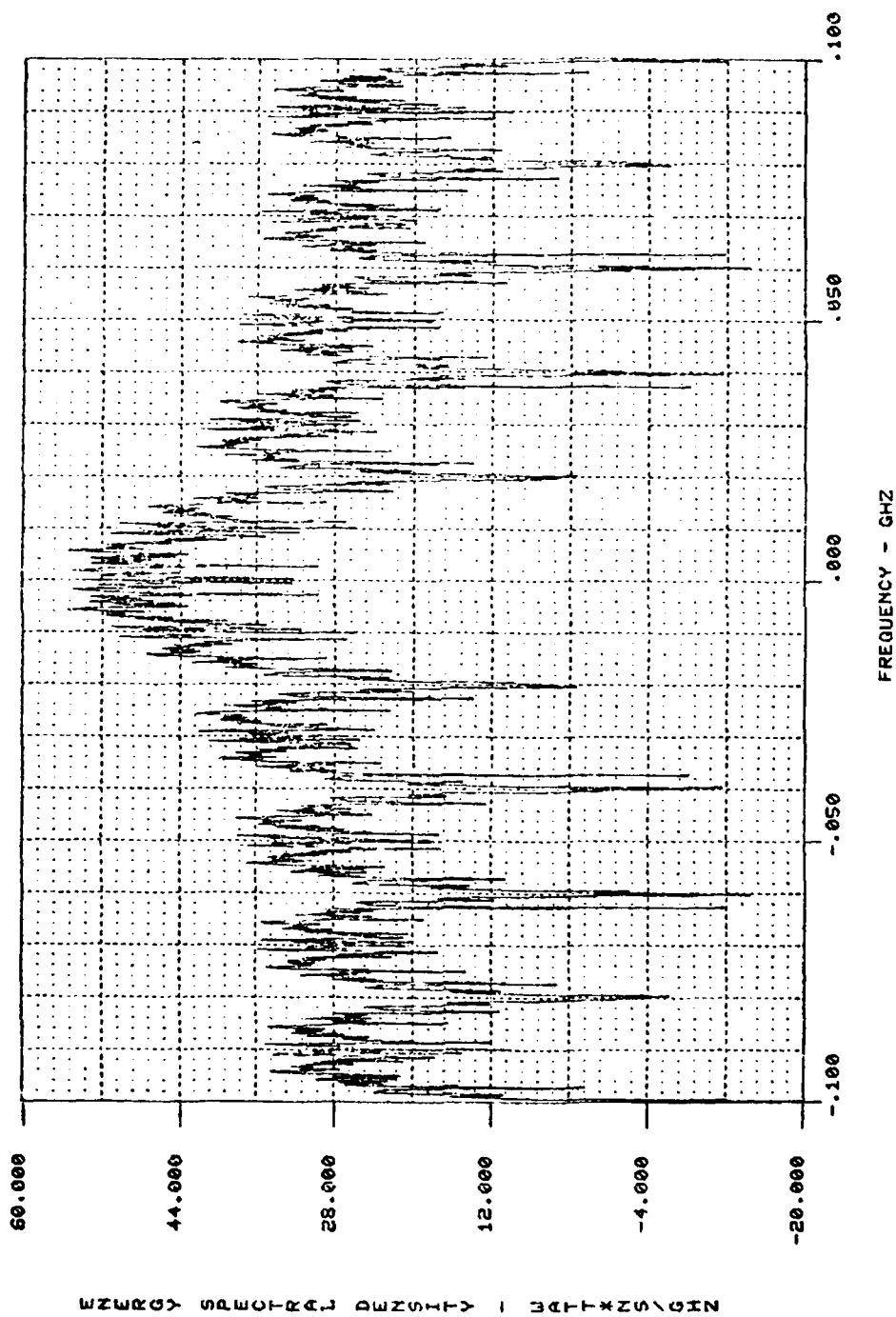


Figure C-2 Spectrum of PRBS Waveform

Type CR When Ready

=

Enter CMD

MATFIL

Enter 2 Input Waveform Names

B1 B2

Enter 2 Output Waveform Names

X1 X2

Temporary File Created For: X1

Temporary File Created For: X2

Enter CMD

NOISE

Enter 2 Input/Output Waveform Names

A1 A2

SIGMA = 1.0000E 00,

\$

Enter CMD

PLOTTR

Enter 1 Input Waveform Names

A1

Enter ST ,RNG ,NSKP ,

A 2

Plot Xfer Started

Xfer Complete

= 0

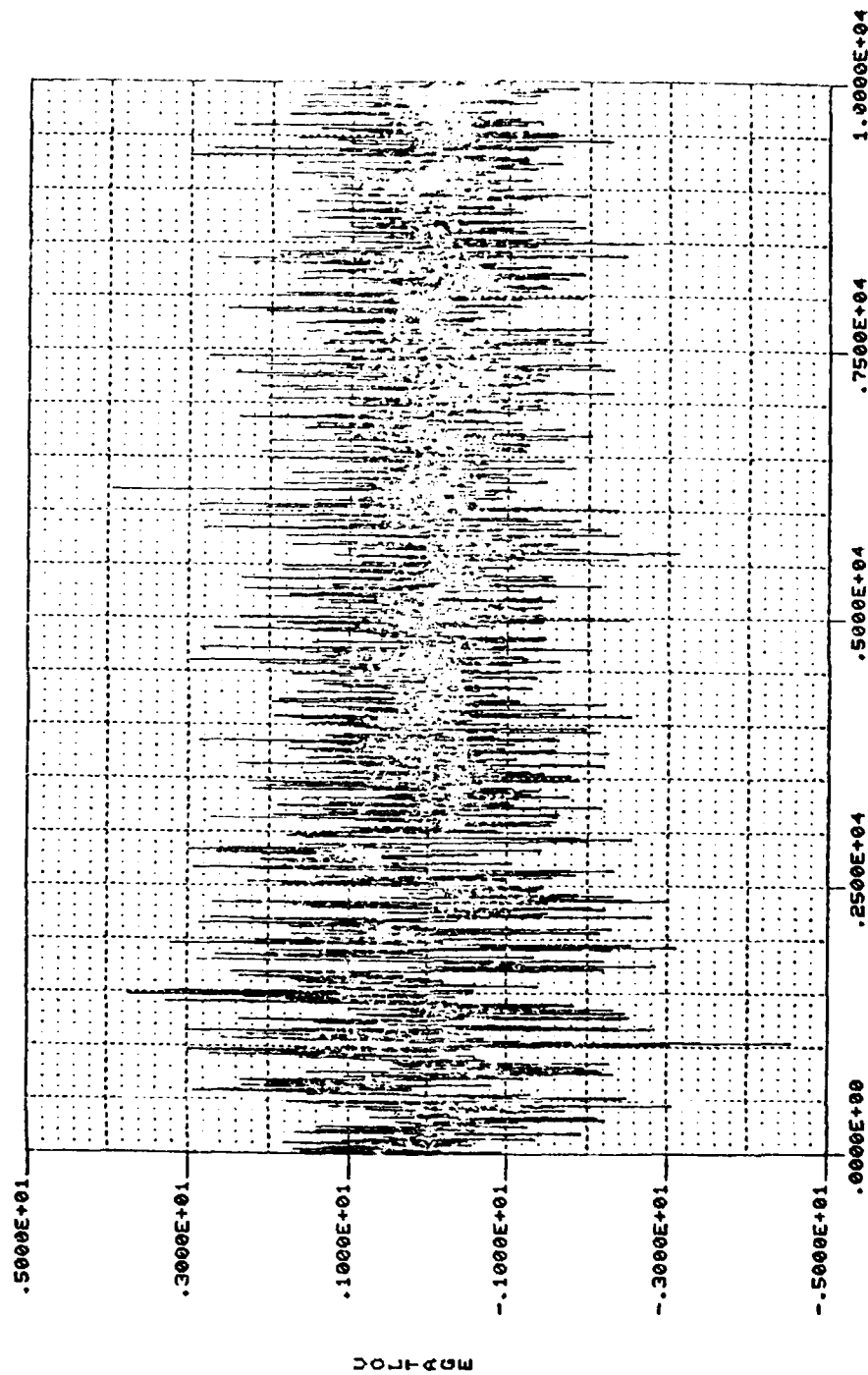


Figure C-3 PRBS Waveform + Noise
(Noise SIGMA = 1.0)

Type CR When Ready

=

Enter CMD

XFRM

Enter 2 Input Waveform Names

A1 A2

Enter 2 Output Waveform Names

B1 B2

Enter CMD

FILT

Enter 2 Input/Output Waveform Names

B1 B2

SF = 3.0000E-02,

FPOLE (1 = -3.0000E-02, 0. ,Z

NP = 1

NZ = 0

\$

Enter CMD

MOVE

Enter 2 Input Waveform Names

X1 X2

Enter 2 Output Waveform Names

A1 A2

Enter CMD

MULT

Enter 2 Input Waveform Names

A1 A2

Enter 2 Input/Output Waveform Names

B1 B2

Enter CMD

IXFRM

Enter 2 Input Waveform Names

B1 B2

Enter 2 Output Waveform Names

A1 A2

Enter CMD

XYTOM

Enter 2 Input Waveform Names

A1 A2

Enter 1 Output Waveform Names

A1

Enter CMD

PLOTTR

Enter 1 Input Waveform Names

A1

Enter ST ,RNG ,NSKP ,

A ,2

Plot Xfer Started

Xfer Complete

= 0

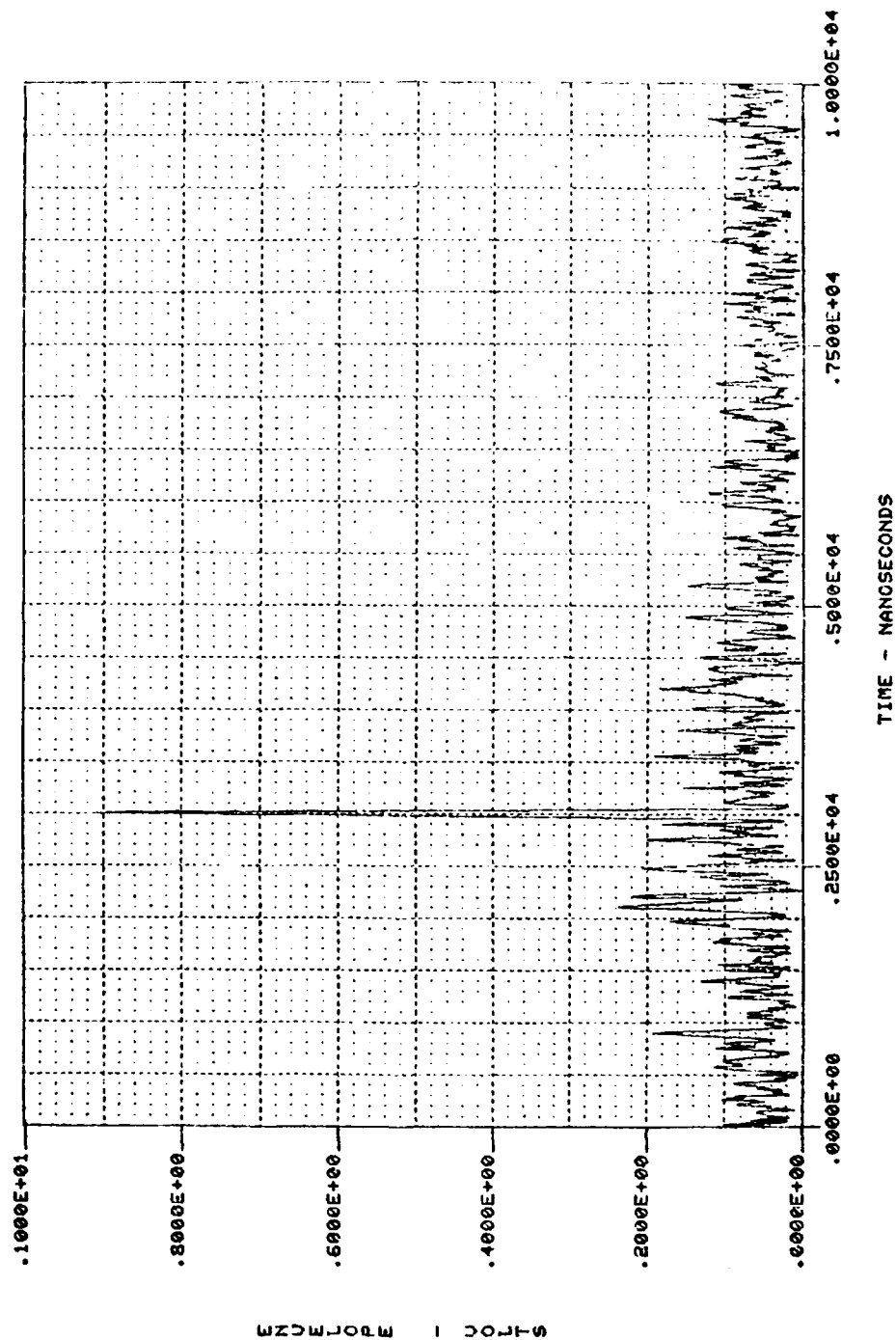


Figure C-4 Matched Filter Output Waveform
(Noise SIGMA = 1.0)

Type CR When Ready

Enter CMD

CLEAR

Enter 1 Output Waveform Names
A2

Enter CMD

XFRM

Enter 2 Input Waveform Names

A1 A2

Enter 2 Output Waveform Names

B1 B2

Enter CMD

FWDET

WARNING: Nonlinear Transfer functions
may generate harmonics which exceed FS/2

Enter 1 Input Waveform Names

A1

Enter 1 Output Waveform Names

A1

Enter CMD

CFAR

WARNING: Nonlinear Transfer functions
may generate harmonics which exceed FS/2

Enter 1 Input Waveform Names

A1

Enter 1 Output Waveform Names

A1

TAUG = 1.0000E 03, THIDL = 1.0000E 02,
\$

Enter CMD

PLOTTR

Enter 1 Input Waveform Names

A1

Enter ST ,RNG ,NSKP ,

A ,2

Plot Xfer Started

Xfer Complete

= 0

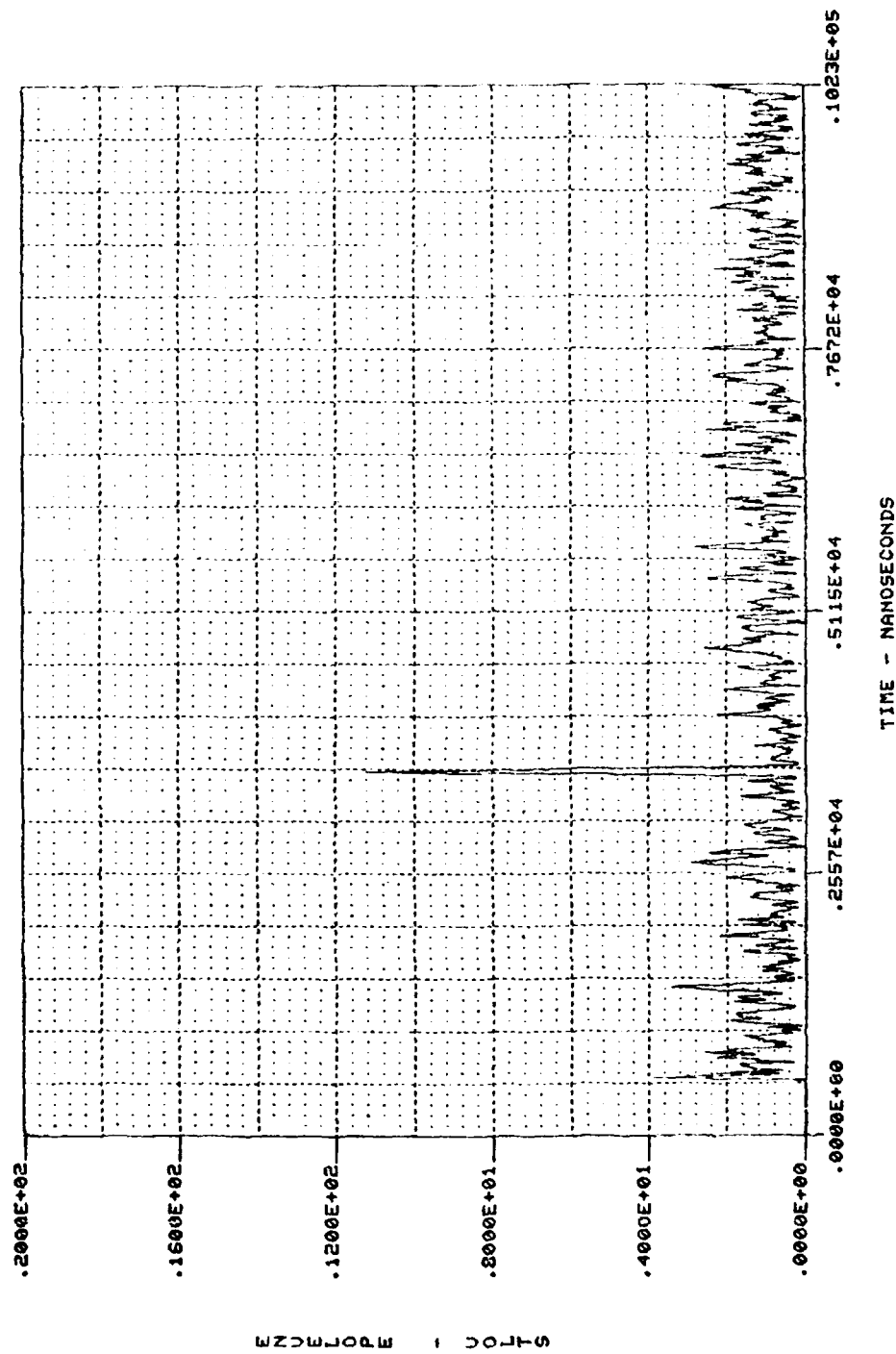


Figure C-5 CFAR Processor Output Waveform
(Noise SIGMA = 1.0)

Type CR When Ready

=

Enter CMD

ENLOOP
Enter CMD

CLOSER

At this point in the example the simulation as defined by the file PRBSCODE had been run. In performing the simulation a Multiple Pass File (MPF) was setup using the commands BGLOOP and ENLOOP. In the listing which follows a parameter of one module was changed and the simulation was rerun using the MPF file. The MPF was printed using the DMP MPF command. Then the parameter SIGMA of the module NOISE was changed using the MODIFY command. Finally the modified MPF was run using the RUN MPF command. In this part of the example hardcopies of some plots were not made since they were the same as those generated earlier.

Enter CMD

^ DMP MPF

1	EGLOOP	315	1	0
2	PCODXY	405	2	0
3	PLOTTR	307	3	0
4	XFRM	219	4	0
5	PLOTDB	306	5	0
6	MATFIL	109	6	0
7	NOISE	402	7	0
8	PLOTTR	307	8	0
9	XFRM	219	9	0
10	FILT	407	10	11
11	MOVE	108	12	0
12	MULT	204	13	0
13	IXFRM	220	14	0
14	XYTOM	105	15	0
15	PLOTTR	307	16	0
16	CLEAR	143	17	0
17	XFRM	219	18	0
18	FWDET	416	19	0
19	CFAR	459	20	0
20	PLOTTR	307	21	0

Enter CMD

^ MODIFY

Enter ISTEP ,
7

Module = NOISE

Enter SIGMA ,
3.0

Enter ISTEP ,

Enter CMD

^ RUN MPF

Module = EGLOOP

Module = PCODXY

Module = PLOTTR

Enter ST ,RNG ,NSKP ,
10.

Plot Xfer Started
Xfer Complete
= 0

AD-A087 562

SIMULATION TECHNOLOGY INC NASHVILLE TN
INCREASED OPERATIONAL FACILITY OF THE IRSS.(U)
MAY 80 R J HANCOCK

F/G 17/9

UNCLASSIFIED

RADC-TR-80-150

F30602-79-C-0043

NL

2 of 2
BIA
0-7562



END
DATE
FILMED
9-80
DTIC

Type CR When Ready

=

Module = XFRM

Module = PLOTDB

Enter ST ,RNG ,NSKP ,
A ,.01

Plot Xfer Started
Xfer Complete
= 0

Type CR When Ready

=

Module = MATFIL

Module = NOISE

Module = PLOTTR

Enter ST ,RNG ,NSKP ,
A ,.2

Plot Xfer Started
Xfer Complete
= 0

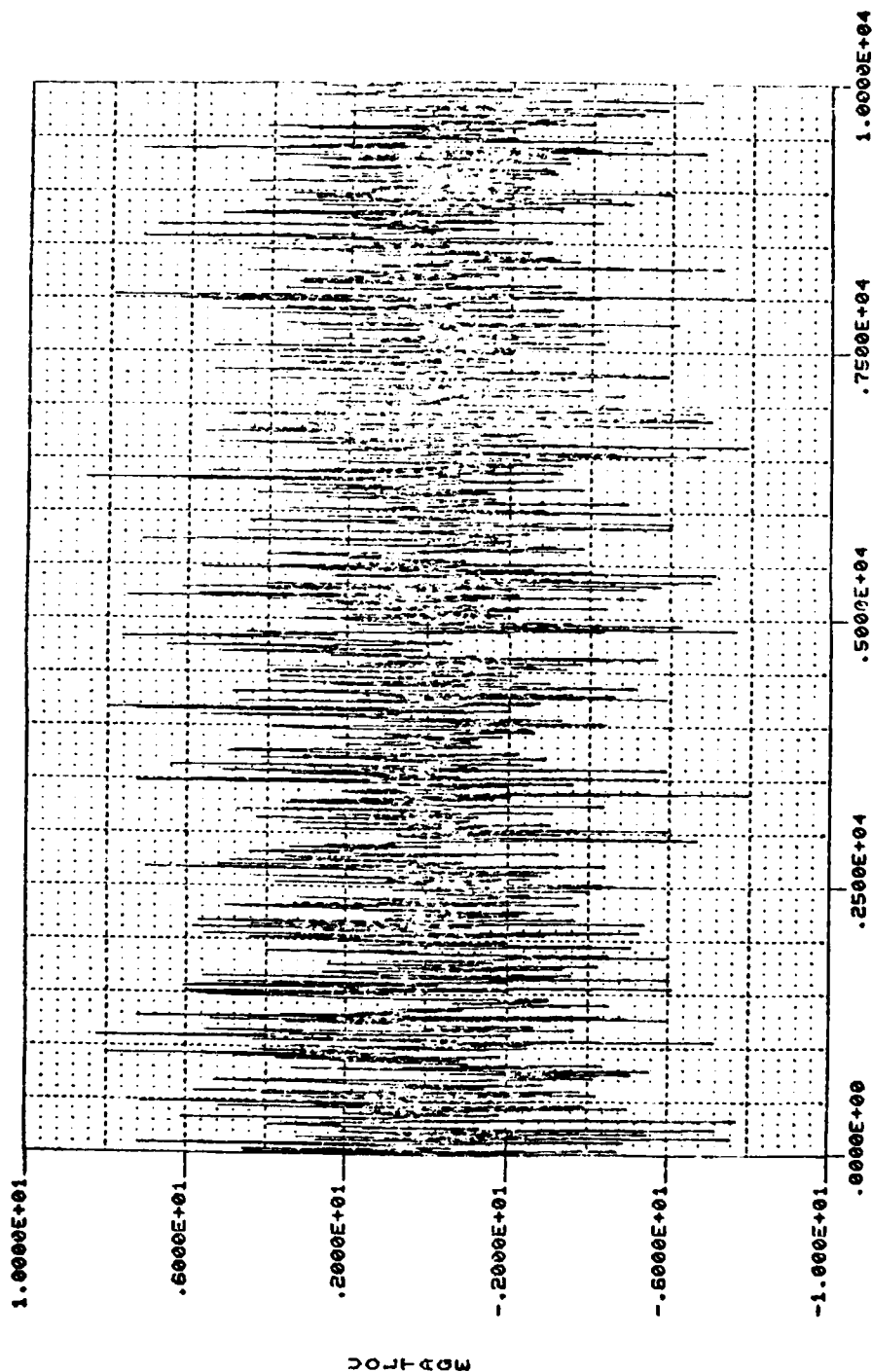


Figure C-6 PRBS Waveform + Noise
(Noise SIGMA = 3.0)

Type CR When Ready

=

Module = XFRM

Module = FILT

Module = MOVE

Module = MULT

Module = IXFRM

Module = XYTOM

Module = PLOTTR

Enter ST ,RNG ,NSKP ,

A ,2

Plot Xfer Started

Xfer Complete

= 0

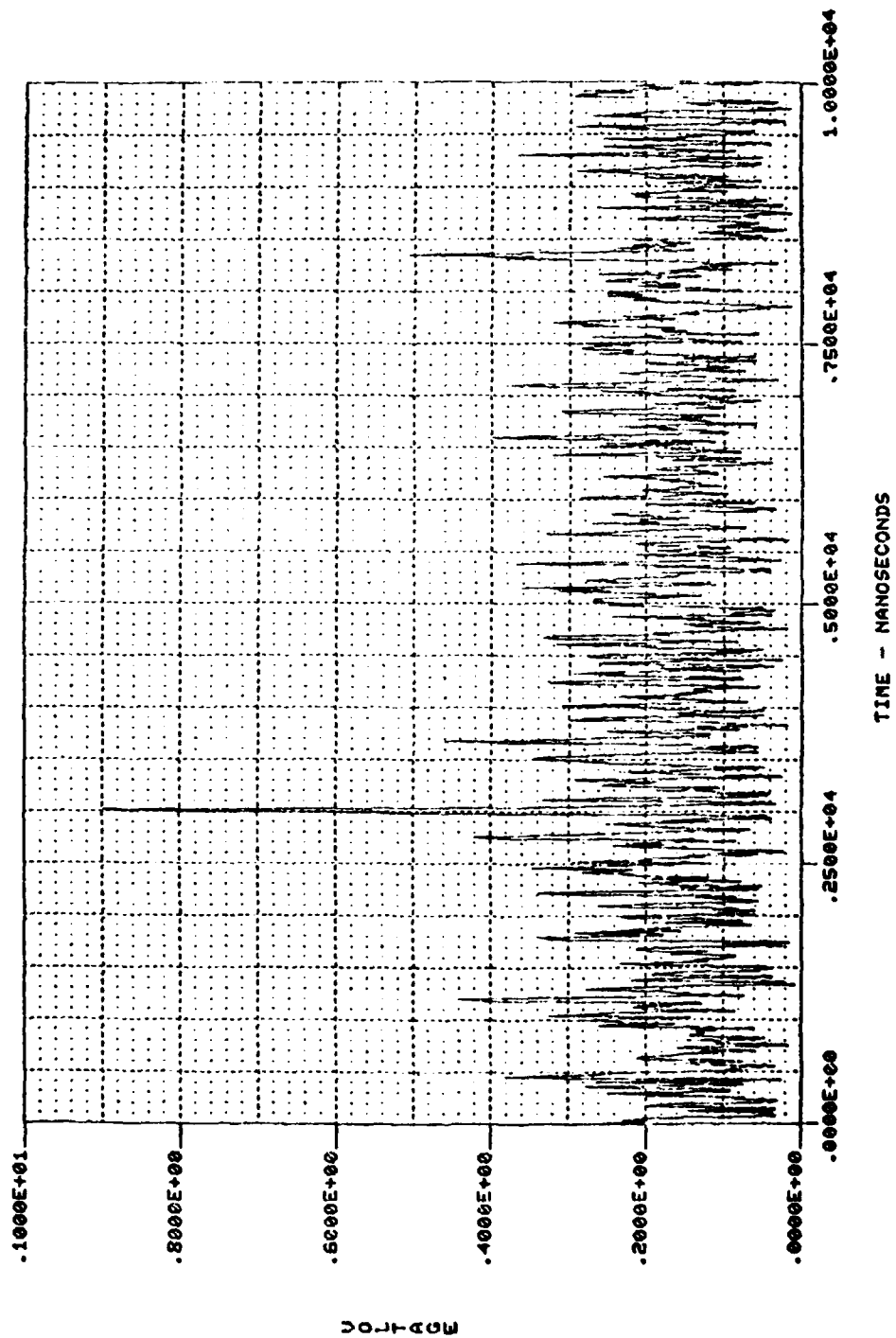


Figure C-7 Matched Filter Output Waveform
(Noise SIGMA = 3.0)

Type CR When Ready
=

Module = CLEAR

Module = XFRM

Module = FWDET

Module = CFAR

Module = PLOTTR

Enter ST ,RNG ,NSKP ,
^ ,2

Plot Xfer Started
Xfer Complete
= 0

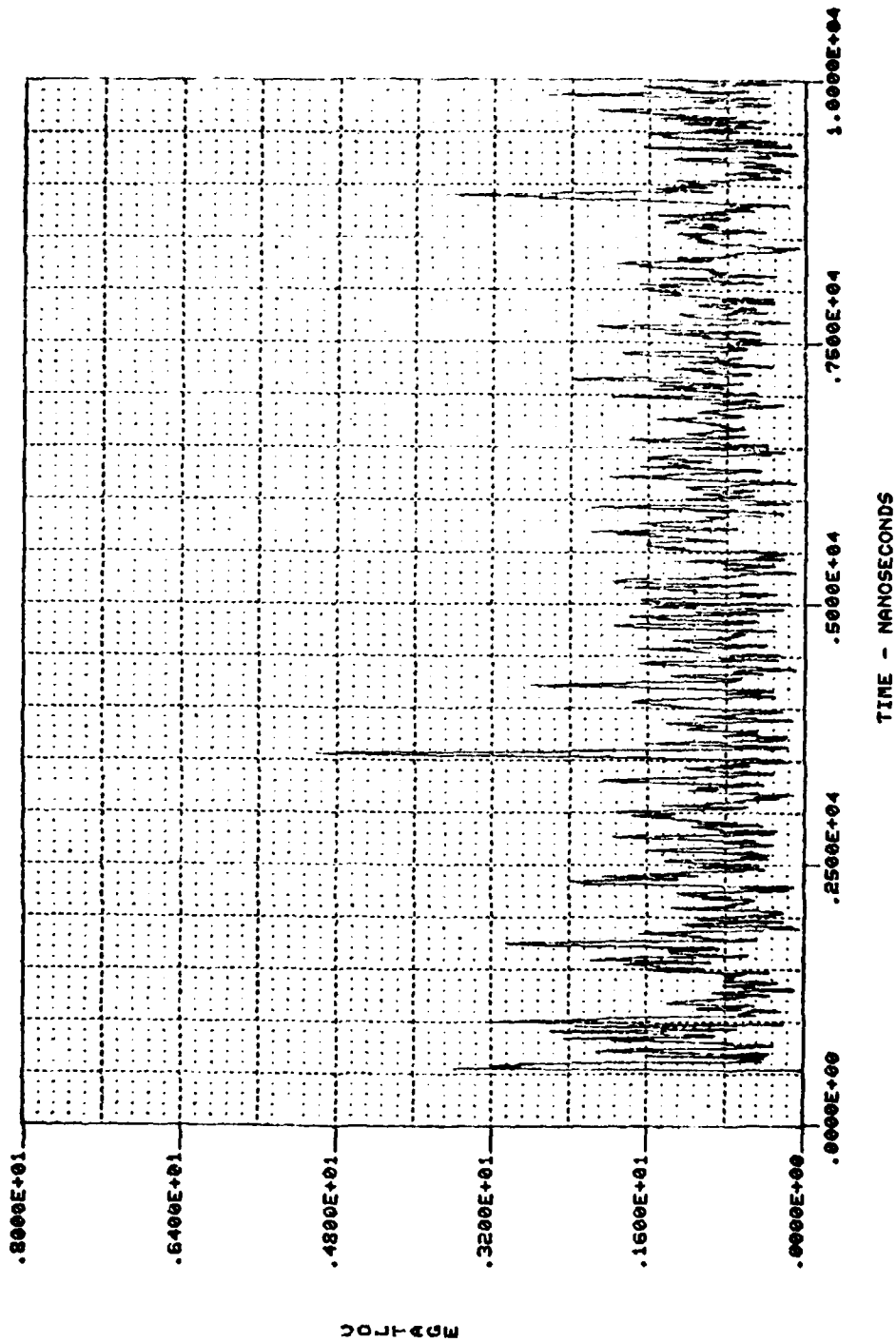


Figure C-8 CFAR Processor Output Waveform
(Noise SIGMA = 3.0)

Type CR When Ready

Enter CMD

^ STOP

IRSS Terminating

* BYE

A P P E N D I X D
L I S T O F A C R O N Y M S

Acronyms which are related to the Honeywell 6180 computer or its operating system are indicated by "(Honeywell)" following the entry. Terms which are obsolete are not included in this table.

BDS	- Block Diagram Subsystem
DUIS	- Dedicated User Interface Subsystem
GCOS	- General Comprehensive Operating Supervisor
GCS	- Graphics Control Subsystem
HSTAR	- Disc File Containing a Load Module (Honeywell)
IFE	- Interactive Front End
IRSS	- Interactive Radar System Simulator
ITS	- Interactive Translator Subsystem
MAT	- Module Array Table
MPF	- Multiple Pass File
MPT	- Module Parameter Table
PRMFL	- Permanent Disc Storage File (Honeywell)
RADSIM	- Radar System Simulation Model
SMEST	- Simulation Module Execution Sequence Table
SRF	- Simulation Run File
TSS	- Time Sharing System (Honeywell)
UAP	- User Aid Processor
WPS	- Waveform Processing Subsystem
YFORT	- Time Sharing Fortran (Honeywell)



MISSION of Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.